VIDEO CONFERENCE APP

A
Mini Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

By

MAMILLAPALLY. HARI SAI

1602-20-748-012

ALLA NITHIN SREEROOP REDDY

1602-20-748-019

Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2022-23

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

DECLARATION BY THE CANDIDATE

We **MAMILLAPALLY.HARI SAI, ALLA.NITHIN SREEROOP REDDY** bearing hall ticket numbers **1602-20-748-012, 1602-20-748-019** hereby declare that the project report entitled "**VIDEO CONFERENCE APP"** Department of Computer Science & Engineering, VCE, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering**.

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

MAMILLAPALLY. HARI SAI (1602-20-748-012),

ALLA. NITHIN SREEROOPREDDY (1602-20-748-019).

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"VIDEO CONFERENCE APP "** being submitted by **MAMILLAPALLY. HARI SAI, ALLA.NITHIN SREEROOP REDDY** bearing **1602-20-748-012, 1602-20-748-019** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by him/her under my guidance.

Dr.M.JITENDER,	**Dr. T. Adilakshmi,**
Assistant Professor,	**Professor & HOD,**
Internal Guide.	**Dept. of CSE.**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people made it possible and whose encouragement and guidance have made our efforts with success.

We are indebted to the **Internal Guide, Dr.M.JITENDAR**, ASSISTANT PROFESSOR, Computer Science & Engineering, VASAVI COLLEGE OF ENGINEERING, Ibrahimbagh, hyd-31 for her support and guidance throughout the project.

We are also indebted to the **Head of the Department, Dr.T.Adilakshmi**, PROFESSOR, Computer Science and Engineering, VASAVI COLLEGE OF ENGINEERING, Ibrahimbagh, hyd-31 for her support and guidance throughout the project.

We extend our deep sense of gratitude to the **Principal, S.V.RAMANA RAO**, VASAVI COLLEGE OF ENGINEERING, Ibrahimbagh, hyd-31 permitting us to undertake this project

ABSTRACT

Video conferencing plays vital role in communication which can be point-to-point or multipoint real-time communication between two or more student located at different geographical locations. However, it is very difficult to adjust a video capturing device to focus on faculty, especially in distance education classroom where faculty moves off the camera while delivering lecture. Also in distance education, there is requirement of real-time interaction between faculty and student, getting feedback of student's reactions at remote class to a lecture is very important. For faculty, it is important to control remote classroom camera particularly student's video freely and easily by local control and also it is important for the student located at any one remote classroom to see the interaction video between faculty and student of other remote classroom.

However, there is some delay between camera control operations at a remote classroom and actual video image displayed on a local screen. These delays are due to network delay between sender and receiver and due to video compression. To operate remote camera interactively, user must consider these delays while controlling these devices. This paper proposes a system for identification and tracking of Active participant with the help of Microsoft Kinect sensor, for the tracking of moving faculty on classroom platform. This paper also proposes solutions for the problems related to real-time interaction between faculty and student and real-time camera control. Here we have discussed latest video encoding and protection technique.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

It is only recently that technology has reached a level of stability, usability and affordability which permits its use in real teaching scenarios rather than research projects. The use of video is being hailed as the next advance in electronic communication. Many companies are developing systems to support such concepts as virtual teams, telecommuting, and remote conferencing (Sami, 2008).

Video conferencing has recently become increasingly popular and disperse in the wake of faster and cheaper internet connections and better technologies. Modern standalone video conferencing units provide advanced video and audio quality due to more efficient compression and can function over normal broadband internet connections. Growing processing power and cheaper accessories, such as webcams, have also made it possible to participate in a video conference using dedicated software on a normal personal computer without any expensive special hardware. With the budget stretched to the breaking point, a number of business premises and institutions are settling aside their travel plans and turning to web conferencing in order to save money and time. Video conference participants use either VC system, web-based application or on-premise software to interactively communicate with co-workers, students and others in virtual meetings or classrooms. This approach is easier, cheaper and much more convenient to use while also providing easy access to file sharing and variety of others collaborative services.

With the explosion of bandwidth, the resources are now available to provide more interaction in the virtual classroom via video conferencing. Using the various technologies available for video conferencing, educators can provide a more interactive distance learning experience by delivering real-time, bidirectional video, voice, and data communications to their distance students, rather than just the standard electronic media.

2.EXPLANATION OF PROJECT

2.1. MODULES USED:

2.1.1. MAIN PAGE:

In this method we will be displaying the joining room page where a user can join or create a room and access his microphone and video.

2.1.2. NAME PAGE:

In this method we will be displaying the box or container in which the user can enter their name and join the meeting according to meeting link.

2.1.3.ROOM PAGE:

In this method according to room meeting id given by the user joins and can access all the type of options on that particular page.

2.1.4.VIDEO OFF PHASE:

In this the user can access his video camera by either switching off or on by his own command.

2.1.5. CHAT PAHSE: .

In this phase the user can chat with other persons who are in the meeting presently.

2.1.6 SCREEN SHARING PHASE:

In this phase the user can share the screen to all other attendees present in the particular meeting

2.1.7 WHITEBOARD:

In this function the user can access the whiteboard and can draw anything he wants and can be seen to all participants in that meeting.

2.2. CONCEPTS USED:

2.2.1 Front-End Implementation:

The Front-End implementation can be done with the help of HTML and designing for CSS. This HTML is a user friendly interface along with various buttons and many other fields which makes the web page more interactive and also has the ability to withstand the robust code implementation

Here HTML helped us to connect with the import the packages the js files and also imported the css file for designing the code and depending upon the computer the design changes and the animations.

2.2.2 Back-End Implementation:

2.2.2.1 Socket Programming:

Here socket programming is the major or heart of this project in which socket i/o is used for joining users according to the room id and the connection between the client and the attendee also can share the messages from one connection to all connections.

2.2.2.2 Javascript Server:

The main heart of Back-End is this Java Script code in which it helps to connect to local host of our computer and can run the code without any interactions of other software. Our Code run on this local host server in which it gives us the idea on how our code run on other computer as well.

3.SYSTEM SPECIFICATIONS

3.1. HARDWARE SPECIFICATIONS:

- Processors: Intel Atom® processor or Intel® Core™ i3 processor
- Disk space: 1 GB or more

3.2. SOFTWARE SPECIFICATIONS:

- **Windows:** 7 or newer
- **MAC:** OS X v10.7 or higher
- **VS Code**

4.SAMPLE CODE

4.1 INDEX PAGE

```
<!DOCTYPE html>

<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <title>QuickMeet - Video Call & Chat</title>
   <meta name="description" content="">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="css/style.css">
   <script src="https://kit.fontawesome.com/6510466b6c.js"
crossorigin="anonymous"></script>

</head>

<body>
   <navbar>
      <div class="logo">QuickMeet</div>
   </navbar>
   <!-- <div class="landing"> -->
      <!-- <div class="head-cont unselectable">
         Welcome to QuickMeet!
      </div> -->

      <div class="main">
         <div class="create-join">
            <div class="text"><div class="head">Create Video Meetings in one-click.</div>
            <div class="subtext">No sign ups required. Open Source platform.</div>
            </div>
            <button id="createroom" class="createroom-butt unselectable">Create
Room</button><br>
            <input type="text" name="room" spellcheck="false" placeholder="Enter Room
Code" id="roomcode" class="roomcode"><br>
            <div class="joinroom unselectable" id="joinroom">Join Room</div>
         </div>
         <div class="video-cont">
            <video class="video-self" autoplay muted playsinline></video>
            <div class="settings">
               <div class="device" id="mic"><i class="fas fa-microphone"></i></div>
               <div class="device" id="webcam"><i class="fas fa-video"></i></div>
            </div>
         </div>
      </div>

      <!-- <div class="main-cont">

         <form class="roomform" action="room.html" method="GET">
            <div class="formcont">
```

```
                    <input type="text" name="name" placeholder="Enter your name" id="username"
class="inputtext"><br>

                    <button type="submit" class="butt unselectable" id="joinroom">Join
Room</button>

                </div>
            </form>

        </div> -->

    <!-- </div> -->

    <script src="js/landing.js"></script>
</body>

</html>
```

4.2 ROOM PAGE

```
<!DOCTYPE html>

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>QuickMeet Room</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/style.css">
    <script src="https://kit.fontawesome.com/6510466b6c.js"
crossorigin="anonymous"></script>
    <script>
        const params = new URLSearchParams(location.search);
        if (!params.get("room")) location.href = "/";
    </script>
</head>

<body>
    <div class="overlay" id="overlay">
        <div class="box">
            <div class="head-name">Enter a Name</div>
            <input type="text" class="name-field" placeholder="Type here.." id="name-
field"></input><br>
            <button class="continue-name">Continue</button>

        </div>
    </div>
    <div class="container-room">
        <div class="left-cont">
```

```
      <div class="video-cont-single" id="vcont">
        <div class="video-box">
          <video class="video-frame" id="vd1" autoplay playsinline>
          </video>
          <div class="nametag" id="myname">yourname</div>
          <div class="mute-icon" id="mymuteicon"><i class="fas fa-microphone-
slash"></i></div>
          <div class="video-off" id="myvideooff">Video Off</div>
        </div>
      </div>

      <div class="whiteboard-cont"><canvas id="whiteboard" height="1000"
width="1000"></canvas>
        <div class="colors-cont">
          <div class="black" onclick="setColor('black')"></div>
          <div class="red" onclick="setColor('#e74c3c')"></div>
          <div class="yellow" onclick="setColor('#f1c40f')"></div>
          <div class="green" onclick="setColor('#badc58')"></div>
          <div class="blue" onclick="setColor('#3498db')"></div>
          <div class="orange" onclick="setColor('#e67e22')"></div>
          <div class="purple" onclick="setColor('#9b59b6')"></div>
          <div class="pink" onclick="setColor('#fd79a8')"></div>
          <div class="brown" onclick="setColor('#834c32')"></div>
          <div class="grey" onclick="setColor('gray')"></div>
          <div class="eraser" onclick="setEraser()"><i class="fas fa-eraser"></i></div>
          <div class="clearboard" onclick="clearBoard()"><i class="fas fa-trash-
alt"></i></div>
        </div>
      </div>

      <div class="footer">
        <div class="utils">

          <div class="audio">
            <i class="fas fa-microphone"></i>
          </div>
          <div class="novideo">
            <i class="fas fa-video"></i>
          </div>
          <div class="screenshare tooltip">
            <i class="fas fa-desktop"></i>
            <span class="tooltiptext">Share Screen</span>
          </div>
          <div class="board-icon tooltip">
            <i class="fas fa-chalkboard"></i>
            <span class="tooltiptext">Whiteboard</span>
          </div>
          <div class="cutcall tooltip">
            <i class="fas fa-phone-slash"></i>
            <span class="tooltiptext">Leave Call</span>
```

```
                </div>


            </div>
            <div class="copycode-cont">
                <div class="roomcode"></div>
                <button class="copycode-button" onclick="CopyClassText()">Copy
Code</button>
            </div>
        </div>


    </div>

    <div class="right-cont">
        <div class="head-title">

            <div class="chats"><i class="fas fa-comment-alt mr-1"></i>Chats</div>
            <div class="attendies"><i class="fas fa-users mr-1"></i>Attendies</div>


        </div>

        <div class="chat-cont">

        </div>
        <div class="chat-input-cont">
            <div class="ci-cont"><input type="text" class="chat-input" placeholder="Type chat
here.."></div>
            <div class="ci-send"><button class="chat-send">Send</button></div>
        </div>
    </div>
  </div>
  <script src="/socket.io/socket.io.js"></script>
  <script src="js/room.js"></script>
</body>

</html>
```

4.3 ROOM JS

```
const socket = io();
const myvideo = document.querySelector("#vd1");
const roomid = params.get("room");
let username;
const chatRoom = document.querySelector('.chat-cont');
const sendButton = document.querySelector('.chat-send');
const messageField = document.querySelector('.chat-input');
const videoContainer = document.querySelector('#vcont');
const overlayContainer = document.querySelector('#overlay')
const continueButt = document.querySelector('.continue-name');
const nameField = document.querySelector('#name-field');
const videoButt = document.querySelector('.novideo');
const audioButt = document.querySelector('.audio');
const cutCall = document.querySelector('.cutcall');
const screenShareButt = document.querySelector('.screenshare');
const whiteboardButt = document.querySelector('.board-icon')

//whiteboard js start
const whiteboardCont = document.querySelector('.whiteboard-cont');
const canvas = document.querySelector("#whiteboard");
const ctx = canvas.getContext('2d');

let boardVisisble = false;

whiteboardCont.style.visibility = 'hidden';

let isDrawing = 0;
let x = 0;
let y = 0;
let color = "black";
let drawsize = 3;
let colorRemote = "black";
let drawsizeRemote = 3;

function fitToContainer(canvas) {
    canvas.style.width = '100%';
    canvas.style.height = '100%';
    canvas.width = canvas.offsetWidth;
    canvas.height = canvas.offsetHeight;
}

fitToContainer(canvas);

//getCanvas call is under join room call
socket.on('getCanvas', url => {
    let img = new Image();
    img.onload = start;
    img.src = url;

    function start() {
        ctx.drawImage(img, 0, 0);
    }

    console.log('got canvas', url)
})

function setColor(newcolor) {
    color = newcolor;
    drawsize = 3;
}
```

```
function setEraser() {
   color = "white";
   drawsize = 10;
}

//might remove this
function reportWindowSize() {
   fitToContainer(canvas);
}

window.onresize = reportWindowSize;
//

function clearBoard() {
   if (window.confirm('Are you sure you want to clear board? This cannot be undone')) {
      ctx.clearRect(0, 0, canvas.width, canvas.height);
      socket.emit('store canvas', canvas.toDataURL());
      socket.emit('clearBoard');
   }
   else return;
}

socket.on('clearBoard', () => {
   ctx.clearRect(0, 0, canvas.width, canvas.height);
})

function draw(newx, newy, oldx, oldy) {
   ctx.strokeStyle = color;
   ctx.lineWidth = drawsize;
   ctx.beginPath();
   ctx.moveTo(oldx, oldy);
   ctx.lineTo(newx, newy);
   ctx.stroke();
   ctx.closePath();

   socket.emit('store canvas', canvas.toDataURL());

}

function drawRemote(newx, newy, oldx, oldy) {
   ctx.strokeStyle = colorRemote;
   ctx.lineWidth = drawsizeRemote;
   ctx.beginPath();
   ctx.moveTo(oldx, oldy);
   ctx.lineTo(newx, newy);
   ctx.stroke();
   ctx.closePath();

}

canvas.addEventListener('mousedown', e => {
    x = e.offsetX;
    y = e.offsetY;
    isDrawing = 1;
})

canvas.addEventListener('mousemove', e => {
    if (isDrawing) {
        draw(e.offsetX, e.offsetY, x, y);
        socket.emit('draw', e.offsetX, e.offsetY, x, y, color, drawsize);
        x = e.offsetX;
        y = e.offsetY;
    }
})

window.addEventListener('mouseup', e => {
    if (isDrawing) {
        isDrawing = 0;
    }
})

socket.on('draw', (newX, newY, prevX, prevY, color, size) => {
    colorRemote = color;
    drawsizeRemote = size;
    drawRemote(newX, newY, prevX, prevY);
})

//whiteboard js end

let videoAllowed = 1;
let audioAllowed = 1;

let micInfo = {};
let videoInfo = {};

let videoTrackReceived = {};

let mymuteicon = document.querySelector("#mymuteicon");
mymuteicon.style.visibility = 'hidden';

let myvideooff = document.querySelector("#myvideooff");
myvideooff.style.visibility = 'hidden';

const configuration = { iceServers: [{ urls: "stun:stun.stunprotocol.org" }] }

const mediaConstraints = { video: true, audio: true };

let connections = {};
let cName = {};
let audioTrackSent = {};
let videoTrackSent = {};
let mystream, myscreenshare;


document.querySelector('.roomcode').innerHTML = `${roomid}`

function CopyClassText() {

    var textToCopy = document.querySelector('.roomcode');
    var currentRange;
    if (document.getSelection().rangeCount > 0) {
        currentRange = document.getSelection().getRangeAt(0);
        window.getSelection().removeRange(currentRange);
    }
    else {
        currentRange = false;
    }

    var CopyRange = document.createRange();
    CopyRange.selectNode(textToCopy);
    window.getSelection().addRange(CopyRange);
    document.execCommand("copy");

    window.getSelection().removeRange(CopyRange);

    if (currentRange) {
        window.getSelection().addRange(currentRange);
    }

    document.querySelector(".copycode-button").textContent = "Copied!"
    setTimeout(()=>{
        document.querySelector(".copycode-button").textContent = "Copy Code";
    }, 5000);
}


continueButt.addEventListener('click', () => {
    if (nameField.value == '') return;
    username = nameField.value;
    overlayContainer.style.visibility = 'hidden';
    document.querySelector("#myname").innerHTML = `${username} (You)`;
    socket.emit("join room", roomid, username);

})

nameField.addEventListener("keyup", function (event) {
    if (event.keyCode === 13) {
        event.preventDefault();
        continueButt.click();
    }
});
```

```
socket.on('user count', count => {
    if (count > 1) {
        videoContainer.className = 'video-cont';
    }
    else {
        videoContainer.className = 'video-cont-single';
    }
})

let peerConnection;

function handleGetUserMediaError(e) {
    switch (e.name) {
        case "NotFoundError":
            alert("Unable to open your call because no camera and/or microphone" +
                "were found.");
            break;
        case "SecurityError":
        case "PermissionDeniedError":
            break;
        default:
            alert("Error opening your camera and/or microphone: " + e.message);
            break;
    }

}


function reportError(e) {
    console.log(e);
    return;
}


function startCall() {

    navigator.mediaDevices.getUserMedia(mediaConstraints)
        .then(localStream => {
            myvideo.srcObject = localStream;
            myvideo.muted = true;

            localStream.getTracks().forEach(track => {
                for (let key in connections) {
                    connections[key].addTrack(track, localStream);
                    if (track.kind === 'audio')
                        audioTrackSent[key] = track;
                    else
                        videoTrackSent[key] = track;
                }
            })
        })
        .catch(handleGetUserMediaError);


}

function handleVideoOffer(offer, sid, cname, micinf, vidinf) {

    cName[sid] = cname;
    console.log('video offered recevied');
    micInfo[sid] = micinf;
    videoInfo[sid] = vidinf;
    connections[sid] = new RTCPeerConnection(configuration);

    connections[sid].onicecandidate = function (event) {
        if (event.candidate) {
            console.log('icecandidate fired');
            socket.emit('new icecandidate', event.candidate, sid);
        }
    };

    connections[sid].ontrack = function (event) {

        if (!document.getElementById(sid)) {
            console.log('track event fired')
            let vidCont = document.createElement('div');
            let newvideo = document.createElement('video');
            let name = document.createElement('div');
            let muteIcon = document.createElement('div');
            let videoOff = document.createElement('div');
            videoOff.classList.add('video-off');
            muteIcon.classList.add('mute-icon');
            name.classList.add('nametag');
            name.innerHTML = `${cName[sid]}`;
            vidCont.id = sid;
            muteIcon.id = `mute${sid}`;
            videoOff.id = `vidoff${sid}`;
            muteIcon.innerHTML = `<i class="fas fa-microphone-slash"></i>`;
            videoOff.innerHTML = 'Video Off'
            vidCont.classList.add('video-box');
            newvideo.classList.add('video-frame');
            newvideo.autoplay = true;
            newvideo.playsinline = true;
            newvideo.id = `video${sid}`;
            newvideo.srcObject = event.streams[0];

            if (micInfo[sid] == 'on')
                muteIcon.style.visibility = 'hidden';
            else
                muteIcon.style.visibility = 'visible';
            if (videoInfo[sid] == 'on')
                videoOff.style.visibility = 'hidden';
            else
                videoOff.style.visibility = 'visible';

            vidCont.appendChild(newvideo);
            vidCont.appendChild(name);
            vidCont.appendChild(muteIcon);
            vidCont.appendChild(videoOff);

            videoContainer.appendChild(vidCont);

        }


    };

    connections[sid].onremovetrack = function (event) {
        if (document.getElementById(sid)) {
            document.getElementById(sid).remove();
            console.log('removed a track');
        }
    };

    connections[sid].onnegotiationneeded = function () {

        connections[sid].createOffer()
            .then(function (offer) {
                return connections[sid].setLocalDescription(offer);
            })
            .then(function () {

                socket.emit('video-offer', connections[sid].localDescription, sid);

            })
            .catch(reportError);
    };

    let desc = new RTCSessionDescription(offer);

    connections[sid].setRemoteDescription(desc)
        .then(() => { return navigator.mediaDevices.getUserMedia(mediaConstraints) })
        .then((localStream) => {

            localStream.getTracks().forEach(track => {
                connections[sid].addTrack(track, localStream);
                console.log('added local stream to peer')
                if (track.kind === 'audio') {
                    audioTrackSent[sid] = track;
                    if (!audioAllowed)
                        audioTrackSent[sid].enabled = false;
                }
                else {
                    videoTrackSent[sid] = track;
                    if (!videoAllowed)
                        videoTrackSent[sid].enabled = false
                }
            })

        })
        .then(() => {
            return connections[sid].createAnswer();
        })
        .then(answer => {
            return connections[sid].setLocalDescription(answer);
        })
        .then(() => {
            socket.emit('video-answer', connections[sid].localDescription, sid);
        })
        .catch(handleGetUserMediaError);


}

function handleNewIceCandidate(candidate, sid) {
    console.log('new candidate recieved')
    var newcandidate = new RTCIceCandidate(candidate);

    connections[sid].addIceCandidate(newcandidate)
        .catch(reportError);
}

function handleVideoAnswer(answer, sid) {
    console.log('answered the offer')
    const ans = new RTCSessionDescription(answer);
    connections[sid].setRemoteDescription(ans);
}

//Thanks to (https://github.com/miroslavpejic85) for ScreenShare Code

screenShareButt.addEventListener('click', () => {
    screenShareToggle();
});
let screenshareEnabled = false;
function screenShareToggle() {
    let screenMediaPromise;
    if (!screenshareEnabled) {
        if (navigator.getDisplayMedia) {
            screenMediaPromise = navigator.getDisplayMedia({ video: true });
        } else if (navigator.mediaDevices.getDisplayMedia) {
            screenMediaPromise = navigator.mediaDevices.getDisplayMedia({ video: true });
        } else {
            screenMediaPromise = navigator.mediaDevices.getUserMedia({
                video: { mediaSource: "screen" },
            });
        }
    } else {
        screenMediaPromise = navigator.mediaDevices.getUserMedia({ video: true });
    }
    screenMediaPromise
        .then((myscreenshare) => {
            screenshareEnabled = !screenshareEnabled;
            for (let key in connections) {
                const sender = connections[key]
                    .getSenders()
                    .find((s) => (s.track ? s.track.kind === "video" : false));
                sender.replaceTrack(myscreenshare.getVideoTracks()[0]);
            }
            myscreenshare.getVideoTracks()[0].enabled = true;
            const newStream = new MediaStream([
                myscreenshare.getVideoTracks()[0],
            ]);
            myvideo.srcObject = newStream;
            myvideo.muted = true;
            mystream = newStream;
            screenShareButt.innerHTML = (screenshareEnabled
                ? `<i class="fas fa-desktop"></i><span class="tooltiptext">Stop Share
Screen</span>`
                : `<i class="fas fa-desktop"></i><span class="tooltiptext">Share Screen</span>`
            );
            myscreenshare.getVideoTracks()[0].onended = function() {
                if (screenshareEnabled) screenShareToggle();
            };
        })
        .catch((e) => {
            alert("Unable to share screen:" + e.message);
            console.error(e);
        });
}

socket.on('video-offer', handleVideoOffer);

socket.on('new icecandidate', handleNewIceCandidate);

socket.on('video-answer', handleVideoAnswer);


socket.on('join room', async (conc, cnames, micinfo, videoinfo) => {
    socket.emit('getCanvas');
    if (cnames)
        cName = cnames;

    if (micinfo)
        micInfo = micinfo;

    if (videoinfo)
        videoInfo = videoinfo;


    console.log(cName);
    if (conc) {
        await conc.forEach(sid => {
            connections[sid] = new RTCPeerConnection(configuration);

            connections[sid].onicecandidate = function (event) {
                if (event.candidate) {
                    console.log('icecandidate fired');
                    socket.emit('new icecandidate', event.candidate, sid);
                }
            };

            connections[sid].ontrack = function (event) {

                if (!document.getElementById(sid)) {
                    console.log('track event fired')
                    let vidCont = document.createElement('div');
                    let newvideo = document.createElement('video');
                    let name = document.createElement('div');
                    let muteIcon = document.createElement('div');
                    let videoOff = document.createElement('div');
                    videoOff.classList.add('video-off');
                    muteIcon.classList.add('mute-icon');
                    name.classList.add('nametag');
                    name.innerHTML = `${cName[sid]}`;
                    vidCont.id = sid;
                    muteIcon.id = `mute${sid}`;
                    videoOff.id = `vidoff${sid}`;
                    muteIcon.innerHTML = `<i class="fas fa-microphone-slash"></i>`;
                    videoOff.innerHTML = 'Video Off'
                    vidCont.classList.add('video-box');
                    newvideo.classList.add('video-frame');
                    newvideo.autoplay = true;
                    newvideo.playsinline = true;
                    newvideo.id = `video${sid}`;
                    newvideo.srcObject = event.streams[0];

                    if (micInfo[sid] == 'on')
                        muteIcon.style.visibility = 'hidden';
                    else
                        muteIcon.style.visibility = 'visible';

                    if (videoInfo[sid] == 'on')
                        videoOff.style.visibility = 'hidden';
                    else
```

```
                    videoOff.style.visibility = 'visible';

                vidCont.appendChild(newvideo);
                vidCont.appendChild(name);
                vidCont.appendChild(muteIcon);
                vidCont.appendChild(videoOff);

                videoContainer.appendChild(vidCont);

            }

        };

        connections[sid].onremovetrack = function (event) {
            if (document.getElementById(sid)) {
                document.getElementById(sid).remove();
            }
        }

        connections[sid].onnegotiationneeded = function () {

            connections[sid].createOffer()
                .then(function (offer) {
                    return connections[sid].setLocalDescription(offer);
                })
                .then(function () {

                    socket.emit('video-offer', connections[sid].localDescription, sid);

                })
                .catch(reportError);
        };

    });

    console.log('added all sockets to connections');
    startCall();

}
else {
    console.log('waiting for someone to join');
    navigator.mediaDevices.getUserMedia(mediaConstraints)
        .then(localStream => {
            myvideo.srcObject = localStream;
            myvideo.muted = true;
            mystream = localStream;
        })
        .catch(handleGetUserMediaError);
}
})
```

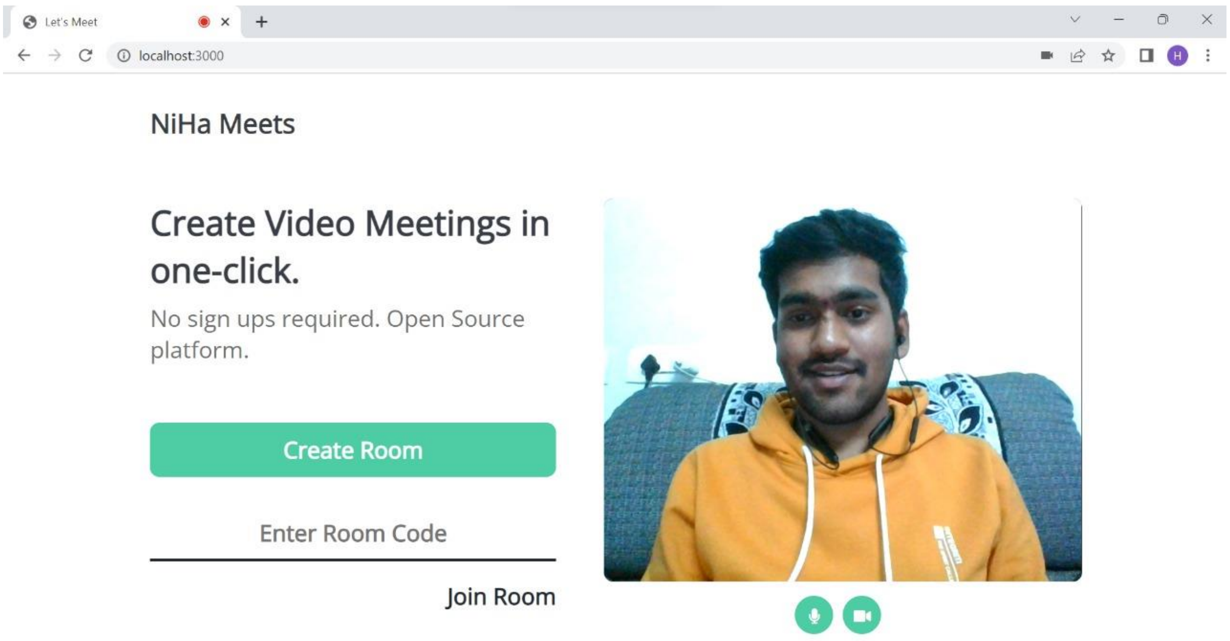
```
socket.on('remove peer', sid => {
    if (document.getElementById(sid)) {
        document.getElementById(sid).remove();
    }

    delete connections[sid];
})

sendButton.addEventListener('click', () => {
    const msg = messageField.value;
    messageField.value = '';
    socket.emit('message', msg, username, roomid);
})

messageField.addEventListener("keyup", function (event) {
    if (event.keyCode === 13) {
        event.preventDefault();
        sendButton.click();
    }
});

socket.on('message', (msg, sendername, time) => {
    chatRoom.scrollTop = chatRoom.scrollHeight;
    chatRoom.innerHTML += `<div class="message">
    <div class="info">
        <div class="username">${sendername}</div>
        <div class="time">${time}</div>
    </div>
    <div class="content">
        ${msg}
    </div>
</div>`
});

videoButt.addEventListener('click', () => {

    if (videoAllowed) {
        for (let key in videoTrackSent) {
            videoTrackSent[key].enabled = false;
        }
        videoButt.innerHTML = `<i class="fas fa-video-slash"></i>`;
        videoAllowed = 0;
        videoButt.style.backgroundColor = "#b12c2c";

        if (mystream) {
            mystream.getTracks().forEach(track => {
                if (track.kind === 'video') {
                    track.enabled = false;
                }
            })
        }
        myvideooff.style.visibility = 'visible';

        socket.emit('action', 'videooff');
    }
    else {
        for (let key in videoTrackSent) {
            videoTrackSent[key].enabled = true;
        }
        videoButt.innerHTML = `<i class="fas fa-video"></i>`;
        videoAllowed = 1;
        videoButt.style.backgroundColor = "#4ECCA3";
        if (mystream) {
            mystream.getTracks().forEach(track => {
                if (track.kind === 'video')
                    track.enabled = true;
            })
        }


        myvideooff.style.visibility = 'hidden';

        socket.emit('action', 'videoon');
    }
})


audioButt.addEventListener('click', () => {

    if (audioAllowed) {
        for (let key in audioTrackSent) {
            audioTrackSent[key].enabled = false;
        }
        audioButt.innerHTML = `<i class="fas fa-microphone-slash"></i>`;
        audioAllowed = 0;
        audioButt.style.backgroundColor = "#b12c2c";
        if (mystream) {
            mystream.getTracks().forEach(track => {
                if (track.kind === 'audio')
                    track.enabled = false;
            })
        }

        mymuteicon.style.visibility = 'visible';

        socket.emit('action', 'mute');
    }
    else {
        for (let key in audioTrackSent) {
            audioTrackSent[key].enabled = true;
        }
        audioButt.innerHTML = `<i class="fas fa-microphone"></i>`;
        audioAllowed = 1;
        audioButt.style.backgroundColor = "#4ECCA3";
        if (mystream) {
            mystream.getTracks().forEach(track => {
                if (track.kind === 'audio')
                    track.enabled = true;
            })
        }

        mymuteicon.style.visibility = 'hidden';

        socket.emit('action', 'unmute');
    }
})

socket.on('action', (msg, sid) => {
    if (msg == 'mute') {
        console.log(sid + ' muted themself');
        document.querySelector(`#mute${sid}`).style.visibility = 'visible';
        micInfo[sid] = 'off';
    }
    else if (msg == 'unmute') {
        console.log(sid + ' unmuted themself');
        document.querySelector(`#mute${sid}`).style.visibility = 'hidden';
        micInfo[sid] = 'on';
    }
    else if (msg == 'videooff') {
        console.log(sid + 'turned video off');
        document.querySelector(`#vidoff${sid}`).style.visibility = 'visible';
        videoInfo[sid] = 'off';
    }
    else if (msg == 'videoon') {
        console.log(sid + 'turned video on');
        document.querySelector(`#vidoff${sid}`).style.visibility = 'hidden';
        videoInfo[sid] = 'on';
    }
})

whiteboardButt.addEventListener('click', () => {
    if (boardVisisble) {
        whiteboardCont.style.visibility = 'hidden';
        boardVisisble = false;
    }
    else {
        whiteboardCont.style.visibility = 'visible';
        boardVisisble = true;
    }
})

cutCall.addEventListener('click', () => {
    location.href = '/';
})
```

4.4 LANDING JS

```
const createButton = document.querySelector("#createroom");
const videoCont = document.querySelector('.video-self');
const codeCont = document.querySelector('#roomcode');
const joinBut = document.querySelector('#joinroom');
const mic = document.querySelector('#mic');
const cam = document.querySelector('#webcam');

let micAllowed = 1;
let camAllowed = 1;

let mediaConstraints = { video: true, audio: true };

navigator.mediaDevices.getUserMedia(mediaConstraints)
    .then(localstream => {
        videoCont.srcObject = localstream;
    })

function uuidv4() {
    return 'xxyxyxxyx'.replace(/[xy]/g, function (c) {
        var r = Math.random() * 16 | 0, v = c == 'x' ? r : (r & 0x3 | 0x8);
        return v.toString(16);
    });
}

const createroomtext = 'Creating Room...';

createButton.addEventListener('click', (e) => {
    e.preventDefault();
    createButton.disabled = true;
    createButton.innerHTML = 'Creating Room';
    createButton.classList = 'createroom-clicked';

    setInterval(() => {
        if (createButton.innerHTML < createroomtext) {
            createButton.innerHTML = createroomtext.substring(0,
createButton.innerHTML.length + 1);
        }
        else {
            createButton.innerHTML = createroomtext.substring(0,
createButton.innerHTML.length - 3);
        }
    }, 500);

    //const name = nameField.value;
    location.href = `/room.html?room=${uuidv4()}`;
});

joinBut.addEventListener('click', (e) => {
    e.preventDefault();
    if (codeCont.value.trim() == "") {
        codeCont.classList.add('roomcode-error');
        return;
    }
    const code = codeCont.value;
    location.href = `/room.html?room=${code}`;
})

codeCont.addEventListener('change', (e) => {
    e.preventDefault();
    if (codeCont.value.trim() !== "") {
        codeCont.classList.remove('roomcode-error');
        return;
    }
})

cam.addEventListener('click', () => {
    if (camAllowed) {
        mediaConstraints = { video: false, audio: micAllowed ? true : false };
        navigator.mediaDevices.getUserMedia(mediaConstraints)
            .then(localstream => {
                videoCont.srcObject = localstream;
            })

        cam.classList = "nodevice";
        cam.innerHTML = `<i class="fas fa-video-slash"></i>`;
        camAllowed = 0;
    }
    else {
        mediaConstraints = { video: true, audio: micAllowed ? true : false };
        navigator.mediaDevices.getUserMedia(mediaConstraints)
            .then(localstream => {
                videoCont.srcObject = localstream;
            })

        cam.classList = "device";
        cam.innerHTML = `<i class="fas fa-video"></i>`;
        camAllowed = 1;
    }
})

mic.addEventListener('click', () => {
    if (micAllowed) {
        mediaConstraints = { video: camAllowed ? true : false, audio: false };
        navigator.mediaDevices.getUserMedia(mediaConstraints)
            .then(localstream => {
                videoCont.srcObject = localstream;
            })

        mic.classList = "nodevice";
        mic.innerHTML = `<i class="fas fa-microphone-slash"></i>`;
        micAllowed = 0;
    }
    else {
        mediaConstraints = { video: camAllowed ? true : false, audio: true };
        navigator.mediaDevices.getUserMedia(mediaConstraints)
            .then(localstream => {
                videoCont.srcObject = localstream;
            })

        mic.innerHTML = `<i class="fas fa-microphone"></i>`;
        mic.classList = "device";
        micAllowed = 1;
    }
})
```

5. OUTPUT SCREENSHOTS AND RESULTS

5.1 Starting Page



Let's Meet

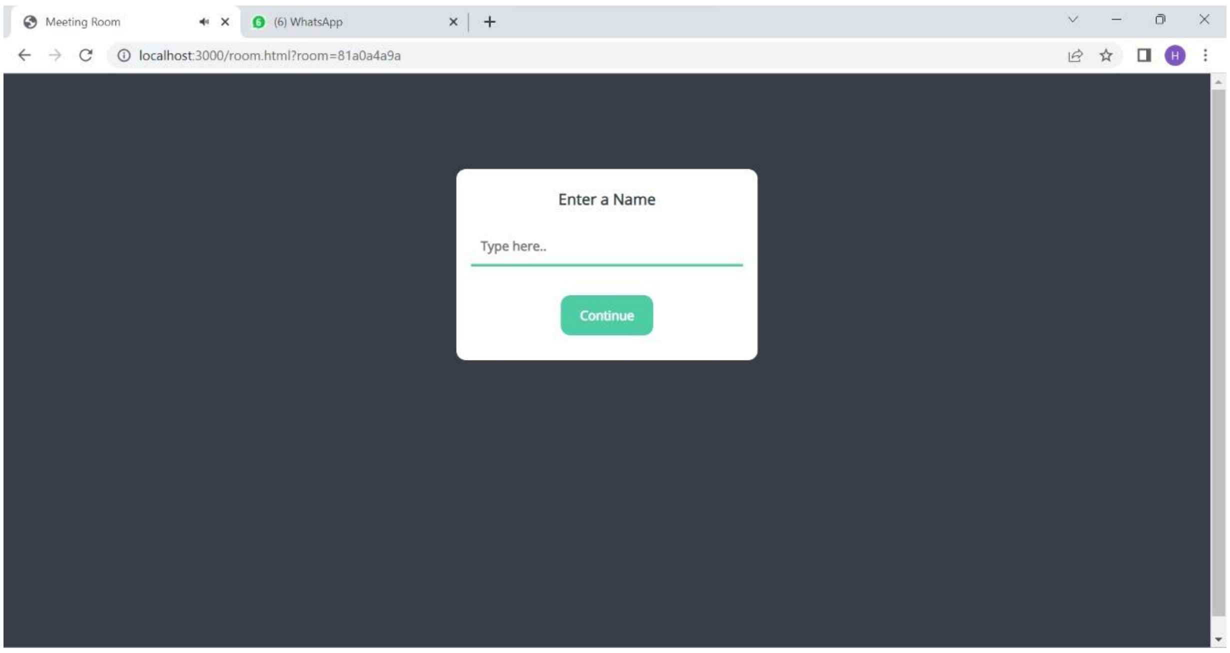
localhost:3000

NiHa Meets

Create Video Meetings in one-click.

No sign ups required. Open Source platform.

5.2 Name Room



Meeting Room

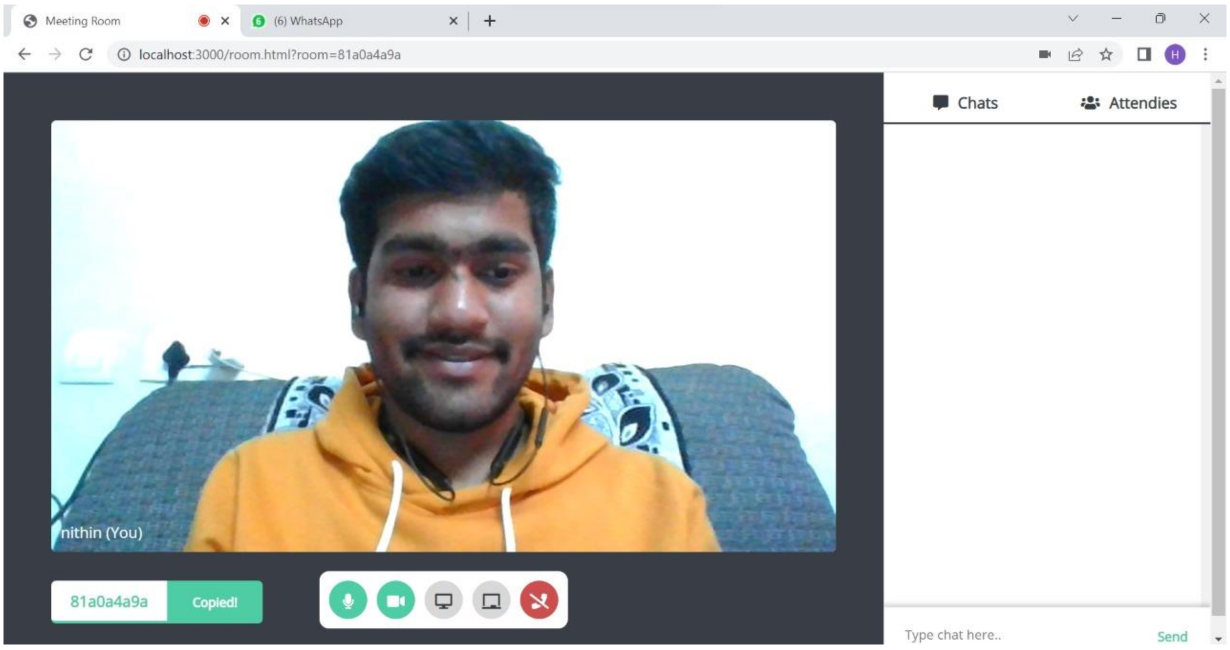
(6) WhatsApp

localhost:3000/room.html?room=81a0a4a9a

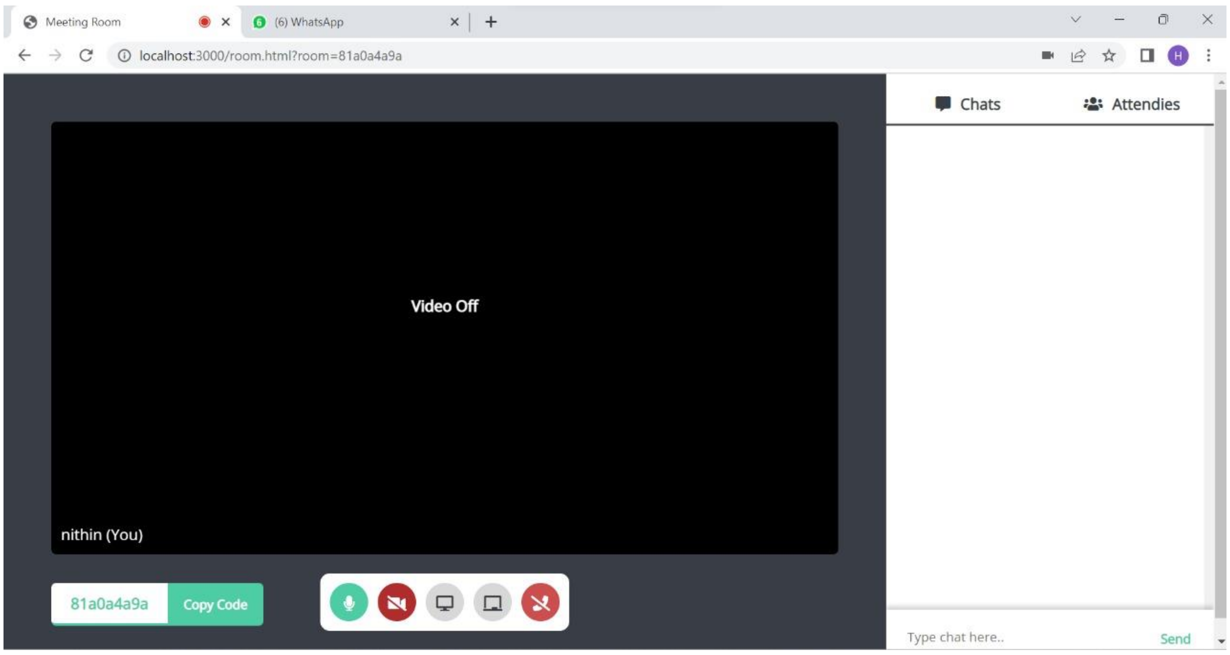
Enter a Name

Type here..

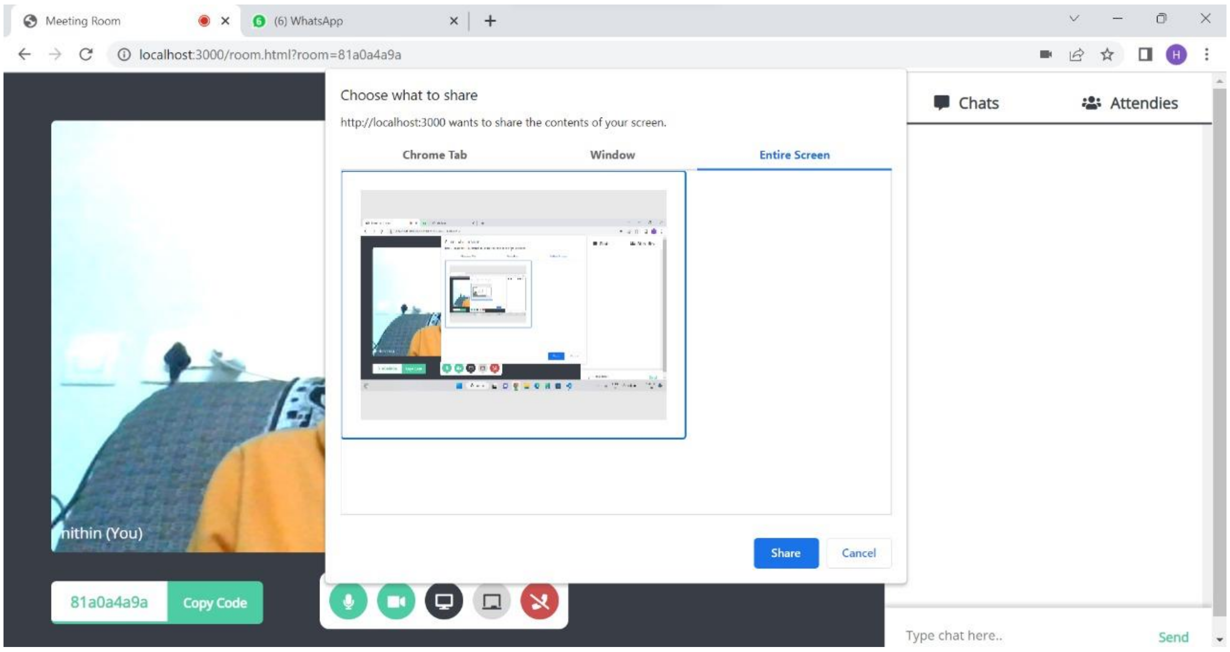
5.3 MEETING ROOM PAGE



5.4 VIDEO OFF PAGE



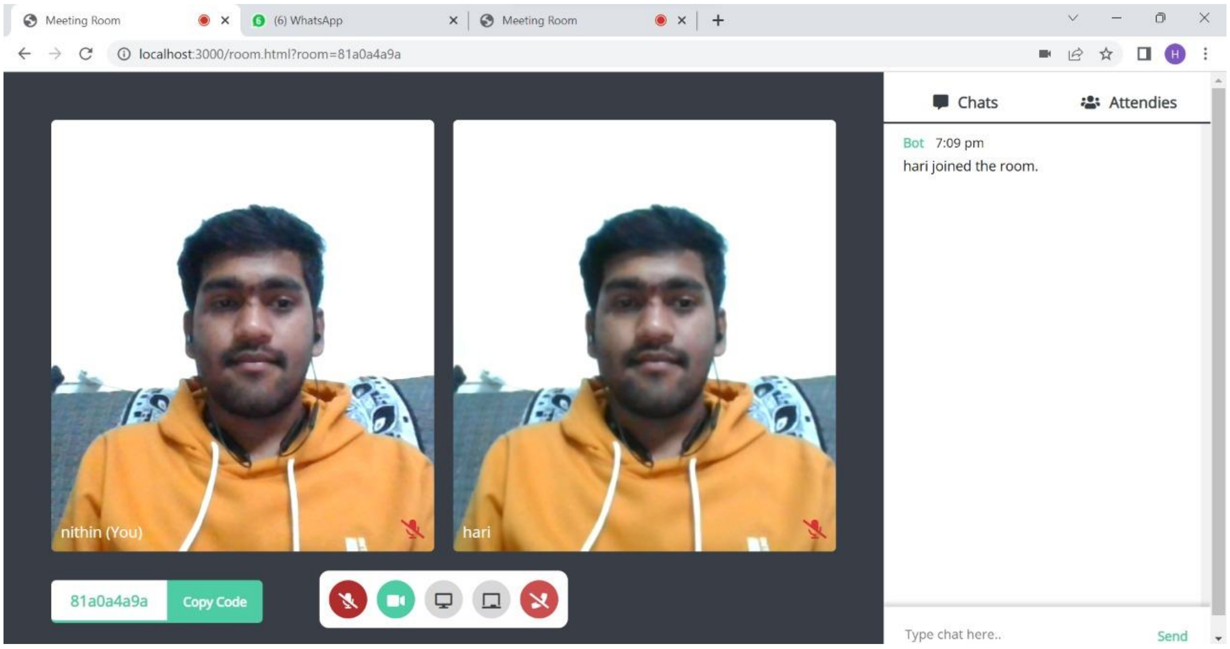
5.5 SHARING SCREEN



Choose what to share

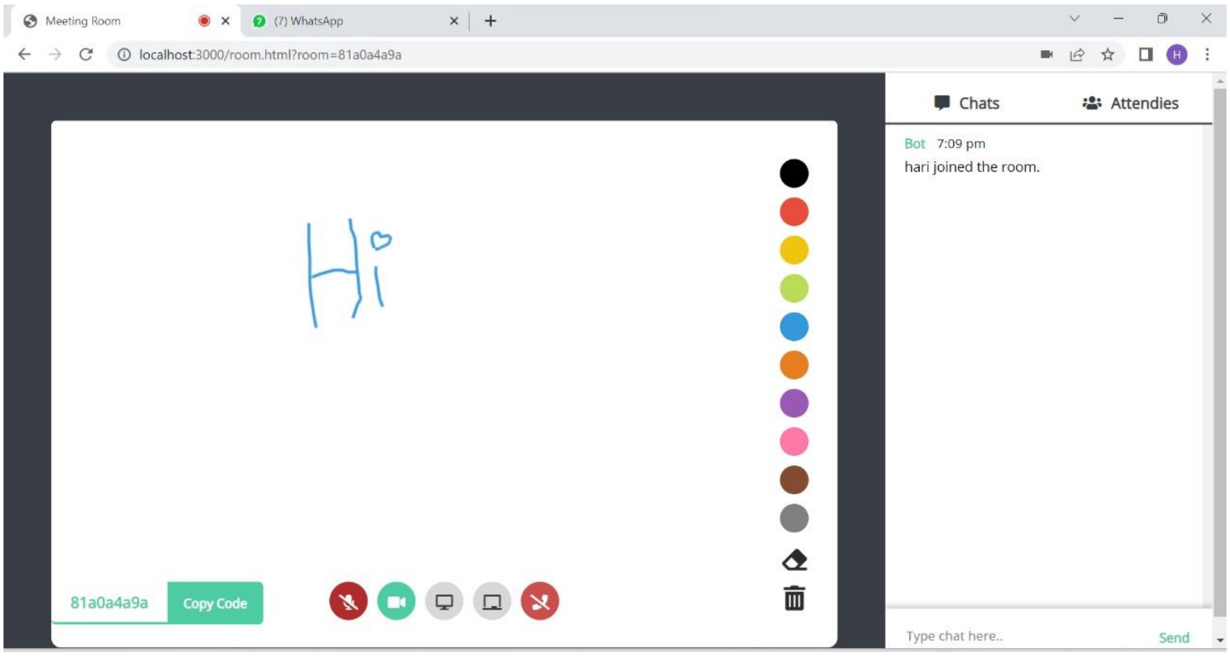
http://localhost:3000 wants to share the contents of your screen.

5.6 MEETING WITH OTHER PERSONS



Bot 7:09 pm
hari joined the room.

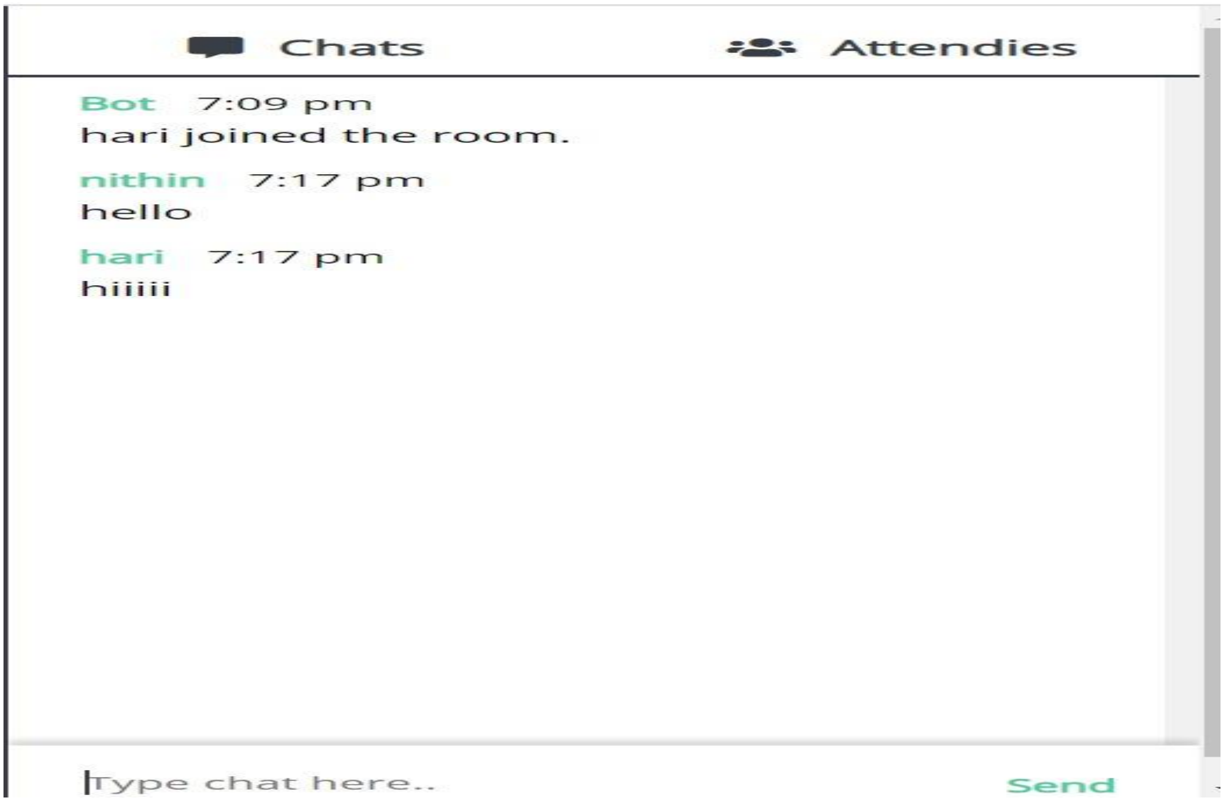
5.7 WHITE BOARD PAGE



Hi

Bot 7:09 pm
hari joined the room.

5.8 CHAT SCREEN



Bot 7:09 pm
hari joined the room.
nithin 7:17 pm
hello
hari 7:17 pm
hiiiii

6.REFERENCES:

[1] **https://socket.io/get-started/chat**

[2] **https://cs.lmu.edu/~ray/notes/jsnetexamples/**

[3] https://www.geeksforgeeks.org/socket-in-computer-network/

[4] **https://javascript.info/websocket#:~:text=To%20open%20a%20websocket%20connection,It's%20like%20HTTPS%20for%20websockets**

[5] **http://www.html5videotutorial.com/video-conferencing**