	Pimpri Chinchwad Education Trust's **Pimpri Chinchwad College of Engineering**

Project Solution

Roll No: FYMCA2021

Name of Student: Kiran Nagnath Chavan **Submission Date:** 30 /07 /2021

Project Title : Web Meeting Appliation

Technologies used : ReactJs, NodeJs, SocketIO, RTCPeerNetworkAPI .

Hosted Link : https://web-meeting-application.herokuapp.com/

Solution:

Server.js

```
const express = require('express')
const http = require('http')
var cors = require('cors')
const app = express()
const bodyParser = require('body-parser')
const path = require("path")
var xss = require("xss")

var server = http.createServer(app)
var io = require('socket.io')(server)

app.use(cors())
app.use(bodyParser.json())

if(process.env.NODE_ENV==='production'){
    app.use(express.static(__dirname+"/build"))
    app.get("*", (req, res) => {
        res.sendFile(path.join(__dirname+"/build/index.html"))
    })
}
app.set('port', (process.env.PORT || 4001))

sanitizeString = (str) => {
    return xss(str)
}
```

```
connections = {}
usernames = {}
messages = {}
timeOnline = {}

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

    socket.on('join-call', (path, username) => {
        if(connections[path] === undefined){
            connections[path] = []
            usernames[path] = []
        }
        console.log("username is : ",username);
        connections[path].push(socket.id)
        usernames[path].push(username)

        timeOnline[socket.id] = new Date()

        for(let a = 0; a < connections[path].length; ++a){
            io.to(connections[path][a]).emit("user-joined", socket.id,
connections[path], usernames[path])
        }

        if(messages[path] !== undefined){
            for(let a = 0; a < messages[path].length; ++a){
                io.to(socket.id).emit("chat-message",
messages[path][a]['data'],
                    messages[path][a]['sender'],
messages[path][a]['socket-id-sender'])
            }
        }

        console.log(path, connections[path])
    })

    socket.on('signal', (toId, message) => {
        io.to(toId).emit('signal', socket.id, message)
    })

    socket.on('chat-message', (data, sender) => {
        data = sanitizeString(data)
        sender = sanitizeString(sender)

        var key
        var ok = false
```

```
        for (const [k, v] of Object.entries(connections)) {
            for(let a = 0; a < v.length; ++a){
                if(v[a] === socket.id){
                    key = k
                    ok = true
                }
            }
        }

        if(ok === true){
            if(messages[key] === undefined){
                messages[key] = []
            }
            messages[key].push({"sender": sender, "data": data,
"socket-id-sender": socket.id})
            console.log("message", key, ":", sender, data)

            for(let a = 0; a < connections[key].length; ++a){
                io.to(connections[key][a]).emit("chat-message", data,
sender, socket.id)
            }
        }
    })

    socket.on('disconnect', () => {
        var diffTime = Math.abs(timeOnline[socket.id] - new Date())
        var key
        for (const [k, v] of
JSON.parse(JSON.stringify(Object.entries(connections)))) {
            for(let a = 0; a < v.length; ++a){
                if(v[a] === socket.id){
                    key = k

                    for(let a = 0; a < connections[key].length; ++a){
                        io.to(connections[key][a]).emit("user-left",
socket.id)
                    }

                    var index = connections[key].indexOf(socket.id)
                    connections[key].splice(index, 1)

                    console.log(key, socket.id, Math.ceil(diffTime / 1000))

                    if(connections[key].length === 0){
                        delete connections[key]
```

```
                    }
                }
            }
        }
    })
})

server.listen(app.get('port'), () => {
    console.log("listening on", app.get('port'))
})
```

App.js

```
import React, { Component } from 'react'
import Video from './Components/Video/Video'
import Home from './Components/Home/Home'
import { BrowserRouter as Router, Switch, Route } from 'react-router-dom';
import './App.css'

class App extends Component {
    render() {
        return (
            <div className="app">
                <Router>
                    <Switch>
                        <Route path="/" exact component={Home} />
                        <Route path="/:url" component={Video} />
                    </Switch>
                </Router>
            </div>
        )
    }
}

export default App;
```

App.css

```

html {
    --section-background-color: linear-gradient(
      to bottom left,
```

```
    rgba(17, 16, 16, 0.582),
    rgba(12, 8, 24, 0.904)
  );
}
 /* #f50057 */
 .app{
  height: 100vh;
  overflow-y: hidden;
  overflow-x: hidden;
  background-color: #202124;
  color: whitesmoke;
}
```

Home.js

```
import React from 'react'
import { useState } from 'react'
import { Button, Col, Container, Form, InputGroup, Row } from
'react-bootstrap'
import { useHistory } from 'react-router-dom'
import './Home.css'

const img =
"https://thumbs.dreamstime.com/b/view-over-businessman-shoulder-laptop-wher
e-four-multiracial-colleagues-engaged-group-meeting-line-video-conference-c
all-178126113.jpg"

const Home = () => {

  const [code, setCode] = useState("")
  const history = useHistory()


   const NewMeet = () => {
       console.log("clicked")
      let code = Math.random().toString(36).substring(2,10)
      history.push(`/${code}`)
  }

  const JoinMeet = () => {
      history.push(`/${code}`)
  }

  // document.addEventListener('click', NewMeet)
```

```
    return (
        <div className="home">
            <div className="home_left">
                <h1>चला भेटू या...</h1>
                <div className="home_left_bottom">
                    <button onClick={NewMeet} className="new_meeting">New
Meeting</button>
                    <input onChange={e=> setCode(e.target.value)}
className="meeting_code" type="text" placeholder="Enter the meeting
code..." />
                    <button onClick={JoinMeet}
className="join_meeting">Join</button>
                </div>

            </div>
            <div className="home_right">
            <img src={img}/>
            </div>
        </div>
    )
}

export default Home
```

Home.css

```
.home{
    display: flex;
    color: white;
    overflow-y: hidden;
    overflow-x: hidden;
}

.home_left{
    flex: 1;
    justify-content: start;
    border-color: red 5px;
    padding-top: 15%;
    padding-left: 10%;
}

.home_left > h1 {
    font-size: 300%;
    color: whitesmoke;
```

```
}

.home_left_bottom{
   padding-top: 10%;

}

.new_meeting{
   background: transparent;
   color: white;
   border-color: whitesmoke;
   font-size: 20px;
   height: 30%;
   margin-right: 5%;
   border-radius: 5%;
   padding: 1%;
}

.new_meeting:hover{
   color: black;
   background: white;
}

.meeting_code{
   text-decoration: none;
   background: transparent;
   border-color: whitesmoke;
   color: white;
   height: 40px;
   font-size: 20px
}


.join_meeting{
   background: transparent;
   color: white;
   border: none;
   height: 20%;
   font-size: 20px;
   height: 45px;
}

.join_meeting:hover{
   background-color: white;
   color: black;
```

```
}

.home_right{
   flex: o.6;
   justify-content: end;
   text-align: center;
   /* padding-top: 10%; */
}

.home_right > img {
   height: 1000px;
   width: 1000px;
   /* opacity: 0.3; */
   mask-image: linear-gradient(to left,rgba(0,0,01),rgba(0,0,0,0));
   /* mask-image: linear-gradient(to bottom,rgb(105, 105,
116),rgba(0,0,0,0)); */
   z-index: -1;
   /* border-radius: 20%; */
}

@media screen and (max-width: 600px){
   .home{
       display: inline;
   }
   .new_meeting{
       margin-bottom: 10%;
   }
   .home_right > img {
       padding-top: 30%;
       width: 100%;
       height: 400px;
   }
}
```

Video.js

```
import React, { Component } from 'react'
import io from 'socket.io-client'
import faker from "faker"

import {IconButton, Badge, Input, Button} from '@material-ui/core'
import VideocamIcon from '@material-ui/icons/Videocam'
import VideocamOffIcon from '@material-ui/icons/VideocamOff'
```

```
import MicIcon from '@material-ui/icons/Mic'
import MicOffIcon from '@material-ui/icons/MicOff'
import ScreenShareIcon from '@material-ui/icons/ScreenShare'
import StopScreenShareIcon from '@material-ui/icons/StopScreenShare'
import CallEndIcon from '@material-ui/icons/CallEnd'
import ChatIcon from '@material-ui/icons/Chat'

import {RiMicFill, RiMicOffFill} from 'react-icons/ri'
import {BiVideo, BiVideoOff} from 'react-icons/bi'
import {MdCallEnd, MdScreenShare, MdStopScreenShare, MdPeople, MdChat} from
'react-icons/md'
import {IoMdMic, IoMdMicOff} from 'react-icons/io'
import {BiVideoRecording} from 'react-icons/bi'

import { message } from 'antd'
import 'antd/dist/antd.css'

import { Row } from 'reactstrap'
import Modal from 'react-bootstrap/Modal'
import 'bootstrap/dist/css/bootstrap.css'
import "./Video.css"
import { Col } from 'react-grid-system'

const server_url = "http://localhost:4001"
// const server_url = "https://web-meeting-application.herokuapp.com/"
var connections = {}
const peerConnectionConfig = {
   'iceServers': [
       // { 'urls': 'stun:stun.services.mozilla.com' },
       { 'urls': 'stun:stun.l.google.com:19302' },
   ]
}
var socket = null
var socketId = null
var elms = 0

class Video extends Component {
   constructor(props) {
       super(props)

       this.localVideoref = React.createRef()

       this.videoAvailable = false
       this.audioAvailable = false
```

```
        this.state = {
            video: false,
            audio: false,
            screen: false,
            chatModal: false,
            userModal: false,
            screenAvailable: false,
            messages: [],
            message: "",
            newmessages: 0,
            askForUsername: true,
            username: "",
            usernames: []
        }
        connections = {}

        this.getPermissions()
    }

    getPermissions = async () => {
        try{
            await navigator.mediaDevices.getUserMedia({ video: true })
                .then(() => this.videoAvailable = true)
                .catch(() => this.videoAvailable = false)

            await navigator.mediaDevices.getUserMedia({ audio: true })
                .then(() => this.audioAvailable = true)
                .catch(() => this.audioAvailable = false)

            if (navigator.mediaDevices.getDisplayMedia) {
                this.setState({ screenAvailable: true })
            } else {
                this.setState({ screenAvailable: false })
            }

            if (this.videoAvailable || this.audioAvailable) {
                navigator.mediaDevices.getUserMedia({ video:
this.videoAvailable, audio: this.audioAvailable })
                    .then((stream) => {
                        window.localStream = stream
                        this.localVideoref.current.srcObject = stream
                    })
                    .then((stream) => {})
                    .catch((e) => console.log(e))
            }
```

```
        } catch(e) { console.log(e) }
    }

    getMedia = () => {
        this.setState({
            video: this.videoAvailable,
            audio: this.audioAvailable
        }, () => {
            this.getUserMedia()
            this.connectToSocketServer()
        })
    }

    getUserMedia = () => {
        if ((this.state.video && this.videoAvailable) || (this.state.audio
&& this.audioAvailable)) {
            navigator.mediaDevices.getUserMedia({ video: this.state.video,
audio: this.state.audio })
                .then(this.getUserMediaSuccess)
                .then((stream) => {})
                .catch((e) => console.log(e))
        } else {
            try {
                let tracks =
this.localVideoref.current.srcObject.getTracks()
                tracks.forEach(track => track.stop())
            } catch (e) {}
        }
    }

    getUserMediaSuccess = (stream) => {
        try {
            window.localStream.getTracks().forEach(track => track.stop())
        } catch(e) { console.log(e) }

        window.localStream = stream
        this.localVideoref.current.srcObject = stream

        for (let id in connections) {
            if (id === socketId) continue

            connections[id].addStream(window.localStream)

            connections[id].createOffer().then((description) => {
                connections[id].setLocalDescription(description)
```

```
                    .then(() => {
                        socket.emit('signal', id, JSON.stringify({ 'sdp':
connections[id].localDescription }))
                    })
                    .catch(e => console.log(e))
            })
        }

        stream.getTracks().forEach(track => track.onended = () => {
            this.setState({
                video: false,
                audio: false,
            }, () => {
                try {
                    let tracks =
this.localVideoref.current.srcObject.getTracks()
                    tracks.forEach(track => track.stop())
                } catch(e) { console.log(e) }

                let blackSilence = (...args) => new
MediaStream([this.black(...args), this.silence()])
                window.localStream = blackSilence()
                this.localVideoref.current.srcObject = window.localStream

                for (let id in connections) {
                    connections[id].addStream(window.localStream)

                    connections[id].createOffer().then((description) => {
                        connections[id].setLocalDescription(description)
                            .then(() => {
                                socket.emit('signal', id, JSON.stringify({
'sdp': connections[id].localDescription }))
                            })
                            .catch(e => console.log(e))
                    })
                }
            })
        })
    }

    getDislayMedia = () => {
        if (this.state.screen) {
            if (navigator.mediaDevices.getDisplayMedia) {
                navigator.mediaDevices.getDisplayMedia({ video: true, audio:
true })
```

```
                    .then(this.getDislayMediaSuccess)
                    .then((stream) => {})
                    .catch((e) => console.log(e))
            }
        }
    }

    getDislayMediaSuccess = (stream) => {
        try {
            window.localStream.getTracks().forEach(track => track.stop())
        } catch(e) { console.log(e) }

        window.localStream = stream
        this.localVideoref.current.srcObject = stream

        for (let id in connections) {
            if (id === socketId) continue

            connections[id].addStream(window.localStream)

            connections[id].createOffer().then((description) => {
                connections[id].setLocalDescription(description)
                    .then(() => {
                        socket.emit('signal', id, JSON.stringify({ 'sdp':
connections[id].localDescription }))
                    })
                    .catch(e => console.log(e))
            })
        }

        stream.getTracks().forEach(track => track.onended = () => {
            this.setState({
                screen: false,
            }, () => {
                try {
                    let tracks =
this.localVideoref.current.srcObject.getTracks()
                    tracks.forEach(track => track.stop())
                } catch(e) { console.log(e) }

                let blackSilence = (...args) => new
MediaStream([this.black(...args), this.silence()])
                window.localStream = blackSilence()
                this.localVideoref.current.srcObject = window.localStream
```

```
                this.getUserMedia()
            })
        })
    }

    gotMessageFromServer = (fromId, message) => {
        var signal = JSON.parse(message)

        if (fromId !== socketId) {
            if (signal.sdp) {
                connections[fromId].setRemoteDescription(new
RTCSessionDescription(signal.sdp)).then(() => {
                    if (signal.sdp.type === 'offer') {

connections[fromId].createAnswer().then((description) => {

connections[fromId].setLocalDescription(description).then(() => {
                                socket.emit('signal', fromId,
JSON.stringify({ 'sdp': connections[fromId].localDescription }))
                            }).catch(e => console.log(e))
                        }).catch(e => console.log(e))
                    }
                }).catch(e => console.log(e))
            }

            if (signal.ice) {
                connections[fromId].addIceCandidate(new
RTCIceCandidate(signal.ice)).catch(e => console.log(e))
            }
        }
    }

    changeCssVideosForMobile = (main) => {
        let widthMain = main.offsetWidth
        let minWidth = "40%"
        if ((widthMain * 30 / 100) < 300) {
            minWidth = "40%"
        }
        let minHeight = "33%"

        let height = String(100 / elms) + "%"
        let width = ""
        if(elms === 0 || elms === 1) {
            width = "100%"
            height = "100%"
```

```
    } else if (elms === 2) {
        width = "100%"
        height = "50%"
    } else if (elms === 3 || elms === 4) {
        width = "100%"
        height = "33%"
    } else {
        height = String(100 / elms) + "%"
    }

    let videos = main.querySelectorAll("video")
    for (let a = 0; a < videos.length; ++a) {
        videos[a].style.minWidth = minWidth
        videos[a].style.minHeight = minHeight
        videos[a].style.borderRadius = "10px"
        videos[a].style.backgroundColor = "#3C4043"
        videos[a].style.setProperty("width", width)
        videos[a].style.setProperty("height", height)
    }

    return {minWidth, minHeight, width, height}
}

changeCssVideos = (main) => {
    let widthMain = main.offsetWidth
    let minWidth = "30%"
    if ((widthMain * 30 / 100) < 300) {
        minWidth = "300px"
    }
    let minHeight = "40%"

    let height = String(100 / elms) + "%"
    let width = ""
    if(elms === 0 || elms === 1) {
        width = "100%"
        height = "100%"
    } else if (elms === 2) {
        width = "45%"
        height = "100%"
    } else if (elms === 3 || elms === 4) {
        width = "35%"
        height = "50%"
    } else {
        width = String(100 / elms) + "%"
    }
```

```
        let videos = main.querySelectorAll("video")
        for (let a = 0; a < videos.length; ++a) {
            videos[a].style.minWidth = minWidth
            videos[a].style.minHeight = minHeight
            videos[a].style.borderRadius = "10px"
            videos[a].style.backgroundColor = "#3C4043"
            videos[a].style.setProperty("width", width)
            videos[a].style.setProperty("height", height)
        }

        return {minWidth, minHeight, width, height}
    }

    connectToSocketServer = () => {
        socket = io.connect(server_url, { secure: true })

        socket.on('signal', this.gotMessageFromServer)

        socket.on('connect', () => {
            socket.emit('join-call', window.location.href,
this.state.username)
            socketId = socket.id

            socket.on('chat-message', this.addMessage)

            socket.on('user-left', (id) => {
                let video = document.querySelector(`[data-socket="${id}"]`)
                if (video !== null) {
                    elms--
                    video.parentNode.removeChild(video)

                    let main = document.getElementById('main')
                    let mq = window.matchMedia("(min-width: 480px)")
                    if(!mq.matches){
                        console.log("mobile")
                        this.changeCssVideosForMobile(main)
                    }else{
                        console.log("desktop")
                        this.changeCssVideos(main)
                    }
                }
            })

            socket.on('user-joined', (id, clients, usernames) => {
```

```
                this.setState({usernames})
                console.log("usernames : ",this.state.usernames)
                clients.forEach((socketListId) => {
                    connections[socketListId] = new
RTCPeerConnection(peerConnectionConfig)
                    // Wait for their ice candidate
                    connections[socketListId].onicecandidate = function
(event) {
                        if (event.candidate != null) {
                            socket.emit('signal', socketListId,
JSON.stringify({ 'ice': event.candidate }))
                        }
                    }

                    // Wait for their video stream
                    connections[socketListId].onaddstream = (event) => {
                        // TODO mute button, full screen button
                        var searchVidep =
document.querySelector(`[data-socket="${socketListId}"]`)
                        if (searchVidep !== null) { // if i don't do this
check it make an empyt square
                            searchVidep.srcObject = event.stream
                        } else {
                            elms = clients.length
                            let main = document.getElementById('main')
                            let cssMesure
                            // let cssMesure = this.changeCssVideos(main)
                            let mq = window.matchMedia("(min-width: 480px)")
                            if(!mq.matches){
                                console.log("mobile1")
                                cssMesure =
this.changeCssVideosForMobile(main)
                            }else{
                                console.log("desktop1")
                                cssMesure = this.changeCssVideos(main)
                            }



                            let video = document.createElement('video')

                            let css = {minWidth: cssMesure.minWidth,
minHeight: cssMesure.minHeight, maxHeight: "100%", margin: "10px",
                                       borderRadius: "10px", backgroundColor
:"#3C4043" , objectFit: "fill"}
```

```
                        for(let i in css) video.style[i] = css[i]

                        video.style.setProperty("width",
cssMesure.width)
                        video.style.setProperty("height",
cssMesure.height)
                        video.setAttribute('data-socket', socketListId)
                        video.srcObject = event.stream
                        video.autoplay = true
                        video.playsinline = true

                        main.appendChild(video)
                    }
                }

                // Add the local video stream
                if (window.localStream !== undefined &&
window.localStream !== null) {

connections[socketListId].addStream(window.localStream)
                } else {
                    let blackSilence = (...args) => new
MediaStream([this.black(...args), this.silence()])
                    window.localStream = blackSilence()

connections[socketListId].addStream(window.localStream)
                }
            })

            if (id === socketId) {
                for (let id2 in connections) {
                    if (id2 === socketId) continue

                    try {
                        connections[id2].addStream(window.localStream)
                    } catch(e) {}

                    connections[id2].createOffer().then((description) =>
{

connections[id2].setLocalDescription(description)
                            .then(() => {
                                socket.emit('signal', id2,
JSON.stringify({ 'sdp': connections[id2].localDescription }))
                            })
```

```
                                .catch(e => console.log(e))
                        })
                    }
                }
            })
        })
    }

    silence = () => {
        let ctx = new AudioContext()
        let oscillator = ctx.createOscillator()
        let dst = oscillator.connect(ctx.createMediaStreamDestination())
        oscillator.start()
        ctx.resume()
        return Object.assign(dst.stream.getAudioTracks()[0], { enabled:
false })
    }
    black = ({ width = 640, height = 480 } = {}) => {
        let canvas = Object.assign(document.createElement("canvas"), {
width, height })
        canvas.getContext('2d').fillRect(0, 0, width, height)
        let stream = canvas.captureStream()
        return Object.assign(stream.getVideoTracks()[0], { enabled: false })
    }

    handleVideo = () => this.setState({ video: !this.state.video }, () =>
this.getUserMedia())
    handleAudio = () => this.setState({ audio: !this.state.audio }, () =>
this.getUserMedia())
    handleScreen = () => this.setState({ screen: !this.state.screen }, () =>
this.getDislayMedia())

    handleEndCall = () => {
        try {
            let tracks = this.localVideoref.current.srcObject.getTracks()
            tracks.forEach(track => track.stop())
        } catch (e) {}
        window.location.href = "/"
    }

    openChat = () => this.setState({ chatModal: true, newmessages: 0 })
    closeChat = () => this.setState({ chatModal: false })
    handleMessage = (e) => this.setState({ message: e.target.value })

    openUser = () => this.setState({ userModal: true})
```

```
    closeUser = () => this.setState({ userModal: false})

    addMessage = (data, sender, socketIdSender) => {
        this.setState(prevState => ({
            messages: [...prevState.messages, { "sender": sender, "data":
data }],
        }))
        if (socketIdSender !== socketId) {
            this.setState({ newmessages: this.state.newmessages + 1 })
        }
    }

    handleUsername = (e) => {
        console.log(this.state.username," username updated")
        this.setState({ username: e.target.value })
    }

    sendMessage = () => {
        socket.emit('chat-message', this.state.message, this.state.username)
        this.setState({ message: "", sender: this.state.username })
    }

    copyUrl = () => {
        let text = window.location.href
        if (!navigator.clipboard) {
            let textArea = document.createElement("textarea")
            textArea.value = text
            document.body.appendChild(textArea)
            textArea.focus()
            textArea.select()
            try {
                document.execCommand('copy')
                message.success("Link copied to clipboard!")
            } catch (err) {
                message.error("Failed to copy")
            }
            document.body.removeChild(textArea)
            return
        }
        navigator.clipboard.writeText(text).then(function () {
            message.success("Link copied to clipboard!")
        }, () => {
            message.error("Failed to copy")
        })
    }
```

```
   connect = () =>{
       console.log(this.username," username is here")
       if(this.state.username === null || this.state.username === undefined
|| this.state.username === ""){
           const mandatory = document.getElementById('mandatory')
           mandatory.style.display = "block"
           // alert("Username is Mandatory!")
       }
       else
           this.setState({ askForUsername: false }, () => this.getMedia())
   }


   render() {
       return (
           <div>
               {this.state.askForUsername === true ?
                   <Row className="screen">
                       <Col>
                       <div className="my-video">
                       <video
style={{borderRadius:"10px",backgroundColor:"#3C4043"}} width="100%"
height="100%"  id="my-video" ref={this.localVideoref} autoPlay muted>
                           </video>
                           {/* <div style={{position:"absolute",
display:"flex", marginLeft:"40%", bottom:"20px"}}>
                                       {
                                           this.state.audio ?
                                               <div className="screen_mic"
onClick={this.handleAudio} >
                                               <IoMdMic
className="screen_mic_icon"/>
                                               </div>
                                           :
                                               <div
className="screen_mic_red"  onClick={this.handleAudio} >
                                               <IoMdMicOff
className="screen_mic_icon"/>
                                               </div>
                                       }
                                       {
                                           this.state.video ?
                                               <div
className="screen_video"  onClick={this.handleVideo}>
```

```
                                                <BiVideo
className="screen_video_icon"/>
                                                </div>
                                            :
                                                <div
className="screen_video_red"  onClick={this.handleVideo}>
                                                <BiVideoOff
className="screen_video_icon"/>
                                                </div>
                                        }
                            </div> */}
                        </div>

                        </Col>
                        <Col>
                            <div class="ready-to-join"
style={{textAlign:"center", marginTop:"200px"}}>
                                <h3>Ready to Join</h3>
                                <p id="mandatory"
style={{color:"red",display:"none"}}>Username is Mandatory!</p>
                                <Input
style={{color:"whitesmoke",borderColor:"whitesmoke"}}
                                 value={this.state.username}
                                 required placeholder="Username" onChange={e
=> this.handleUsername(e)} />
                                <Button variant="contained"
className="join_button" onClick={this.connect} style={{
                                    margin:
"20px",backgroundColor:"#1B73E8", color:"whitesmoke", height:"50px",
width:"200px", fontSize:"20px",
                                    borderRadius:"20px",fontWeight:"bold"
}}>Join Meeting</Button>
                            </div>
                        </Col>
                    </Row>
                    :
                    <div>
                        <div className="meetFooter">
                            <Row>
                                <Col md={5}>
                                    <div
style={{display:"flex",marginLeft:"30px",cursor:"pointer"}}>
                                        <h3 style={{
backgroundColor:"black",borderRadius:"10px"}}
onClick={this.copyUrl}>{window.location.href}</h3>
```

```
                                </div>
                            </Col>
                            <Col md={3}>
                                <div className="icons_center">
                                    {
                                        this.state.audio ?
                                            <div className="mic"
onClick={this.handleAudio} >
                                            <IoMdMic
className="mic_icon"/>
                                            </div>
                                        :
                                            <div className="mic_red"
onClick={this.handleAudio} >
                                            <IoMdMicOff
className="mic_icon"/>
                                            </div>
                                    }
                                    {
                                        this.state.video ?
                                            <div className="video"
onClick={this.handleVideo}>
                                            <BiVideo
className="video_icon"/>
                                            </div>
                                        :
                                            <div className="video_red"
onClick={this.handleVideo}>
                                            <BiVideoOff
className="video_icon"/>
                                            </div>
                                    }
                                    {
                                        this.state.screen ?
                                            <div
className="screen_share_red"  onClick={this.handleScreen}>
                                            <MdStopScreenShare
className="screen_share_icon" />
                                            </div>
                                        :
                                            <div
className="screen_share"  onClick={this.handleScreen}>
                                            <MdScreenShare
className="screen_share_icon" />
                                            </div>
```

```
                                        }
                                        <div className="end"
onClick={this.handleEndCall}>
                                            <MdCallEnd className="end_icon"
/>
                                        </div>
                                    </div>
                                </Col>
                                <Col mc={3}>
                                <div className="icons_right">
                                    <div className="recording">
                                        <BiVideoRecording/>
                                    </div>
                                    <div className="people"
onClick={this.openUser} >

<MdPeople/><sup>{this.state.usernames.length}</sup>
                                    </div>
                                    <div className="chat"
onClick={this.openChat}>
                                        <MdChat/>
                                    </div>
                                </div>
                                </Col>
                            </Row>
                        </div>
                        {/* //modal for chat */}
                        <Modal show={this.state.chatModal}
onHide={this.closeChat} style={{ zIndex: "999999" }}>
                            <Modal.Header closeButton>
                                <Modal.Title>Chat Room</Modal.Title>
                            </Modal.Header>
                            <Modal.Body style={{ overflow: "auto",
overflowY: "auto", height: "400px", textAlign: "left" }} >
                                {this.state.messages.length > 0 ?
this.state.messages.map((item, index) => (
                                    <div key={index} style={{textAlign:
"left"}}>
                                        <p style={{ wordBreak: "break-all"
}}><b>{item.sender}</b>: {item.data}</p>
                                    </div>
                                )) : <p>No message yet</p>}
                            </Modal.Body>
                            <Modal.Footer className="div-send-msg">
```

```
                                <Input placeholder="Message"
value={this.state.message} onChange={e => this.handleMessage(e)} />
                                <Button variant="contained" color="primary"
onClick={this.sendMessage}>Send</Button>
                            </Modal.Footer>
                        </Modal>
                        {/* //modal for list of participants */}
                        <Modal show={this.state.userModal}
onHide={this.closeUser} style={{ zIndex: "999999" }}>
                            <Modal.Header closeButton>
                                <Modal.Title>People List</Modal.Title>
                            </Modal.Header>
                            <Modal.Body style={{ overflow: "auto",
overflowY: "auto", height: "400px", textAlign: "left" }} >
                                {this.state.usernames.length > 0 ?
this.state.usernames.map((item, index) => (
                                    <div key={index} style={{textAlign:
"left"}}>
                                        <p style={{ wordBreak: "break-all"
}}>{index+1} : <b>{item}</b></p><hr></hr>
                                    </div>
                                )) : <p></p>}
                            </Modal.Body>
                        </Modal>


                        <div className="container">
                            <Row id="main" className="flex-container"
style={{ margin: "", padding: "20px" }}>
                                <video id="my-video"
ref={this.localVideoref} autoPlay muted style={{ borderRadius:"10px",
backgroundColor:"#3C4043",
                                    margin: "10px",objectFit: "fill",
                                    width: "100%",height: "100%"}}></video>
                            </Row>
                        </div>
                    </div>
                }
            </div>
        )
    }
}

export default Video
```

Video.css

```
body {
   margin: 0;
}

.flex-container {
   height: calc(100% - 150px);
   width: 100%;
   /* margin: 0 auto; */
   display: flex;
   flex-direction: row;
   align-items: center;
   justify-content: center;
}

.container {
   height: 100vh;
   width: 100vw;
   max-width: 100vw;
   max-height: 100vh;

   align-items: center;
   justify-content: center;
   text-align: center;
}

.div-send-msg {
   display: flex;
   align-items: center;
   justify-content: center;
   text-align: center;
}

.btn-down {
   z-index: 2;
   position: fixed;
   bottom: 0px;
   width: 100%;
   height: auto;
}

.my-video{
```

```
    position: relative;
    background-color: #202124;
    width: 80%;
    border-radius: 20px;
    margin-left: 20%;
}

@media only screen and (max-width : 480px) {
    .my-video{
        position: relative;
        background-color: #202124;
        /* width: 80%; */
        min-width: 300px;
        border-radius: 20px;
        margin: 10%;
        margin-top: 0;
    }
    .screen{
        margin-top: 0;
    }
    .ready-to-join{
        margin-top: 0 !important;
    }
}

.screen{
    margin-top: 100px;
}



/* footer start */

.meetFooter{
    position: fixed;
    bottom: 0;
    align-items: center;
    /* display: flex; */
    width: 100%;
    justify-content: space-between;
}

.meetFooter_left{
    flex: 0.5;
    align-items: center;
}
```

```css
.meetFooter_center{
   /* flex: 0.5; */
   /* margin-left: 20%; */
   /* margin-top: 15px; */
   /* text-align: center;
   align-items: center;
   -webkit-box-align: center;
   box-align: center; */
   align-items: center;
   /* box-pack: start;
   -webkit-box-pack: start;
   justify-content: flex-start;
   flex : 1; */
}


.icons_center{
   display: flex;
   text-align: center;
   /* padding-top: 5%; */
}

.meetFooter_right{
   flex: 0.5;
   text-align: right;
   /* align-items: center; */
}

.mic, .video, .screen_share{
   background-color: #3C4043;
   color: whitesmoke;
   border-radius: 50%;
   width: 40px;
   height: 40px;
   text-align: center;
   margin-right: 5%;
}


.mic_red, .video_red, .screen_share_red{
   background-color: #EA4335;
   color: whitesmoke;
   border-radius: 50%;
   width: 40px;
```

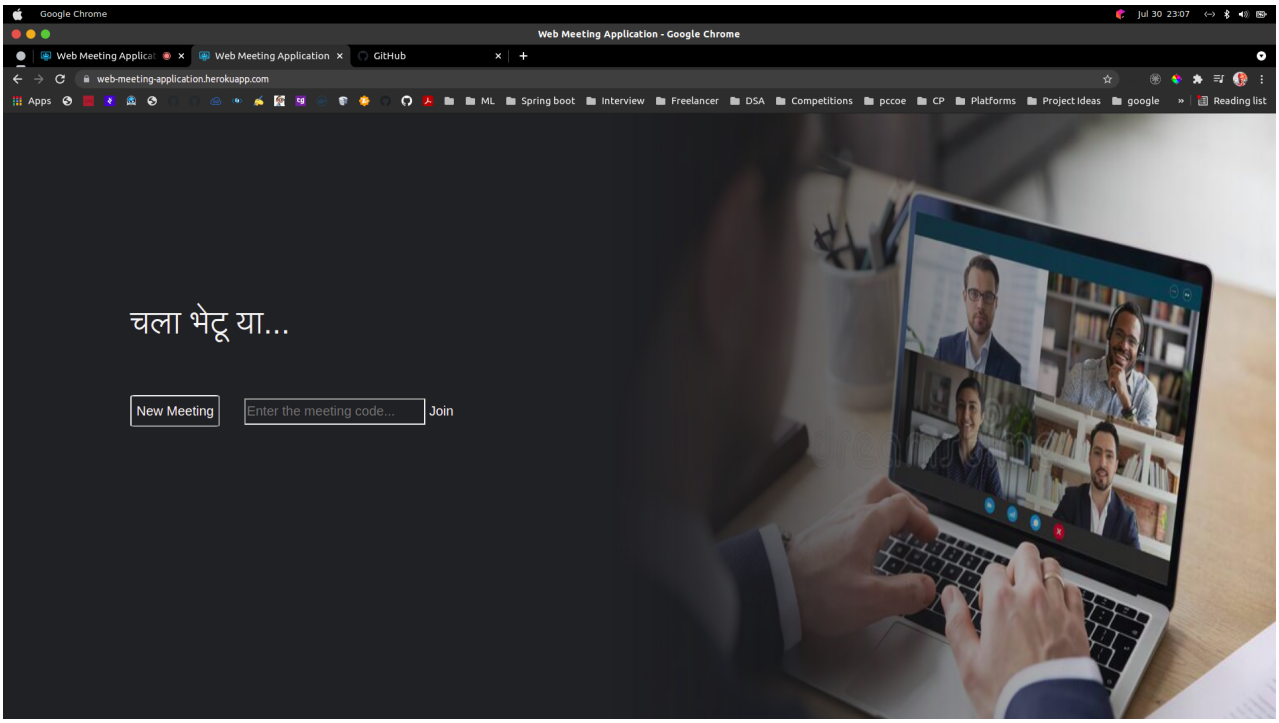
```
    height: 40px;
    text-align: center;
    margin-right: 5%;
}

.mic:hover, .video:hover, .screen_share:hover{
    cursor: pointer;
    background-color: #8a8c8e;
}

.mic_red:hover, .video_red:hover, .screen_share_red:hover{
    cursor: pointer;
    background-color: #ee685d;
}

.mic_icon, .video_icon, .screen_share_icon{
    margin-top: 23%;
    font-size: 20px;
}

.end{
    background-color: #EA4335;
    text-align: center;
    color: whitesmoke;
    border-radius: 100px;
    width: 70px;
    height: 40px;
    text-decoration: none;
    /* margin: 1%; */
}

.end:hover{
    cursor: pointer;
    background-color: #ee685d;
}


.end_icon {
    margin-top: 8%;
    font-size: 25px;
}

h3{
    padding: 1%;
    color: whitesmoke;
```

```
}

.icons_right{
   display: flex;
   position: absolute;
   right: 10px;
   /* padding-top: 5%; */
}

.people, .chat, .recording{
   /* text-align: right; */
   width: 50px;
   height: 40px;
   font-size: 25px;
}

/* footer end  */

/* screen icons  */


.screen_mic, .screen_video{
   /* background-color: #3C4043; */
   background: transparent;
   color: whitesmoke;
   border-radius: 50%;
   border: 1px solid;
   border-color: whitesmoke;
   width: 60px;
   height: 60px;
   text-align: center;
   margin-right: 5%;
   margin-top: 5%;
}


.screen_mic_red, .screen_video_red{
   background-color: #EA4335;
   color: whitesmoke;
   border-radius: 50%;
   border: 1px solid;
   border-color: whitesmoke;
   width: 60px;
   height: 60px;
   text-align: center;
```

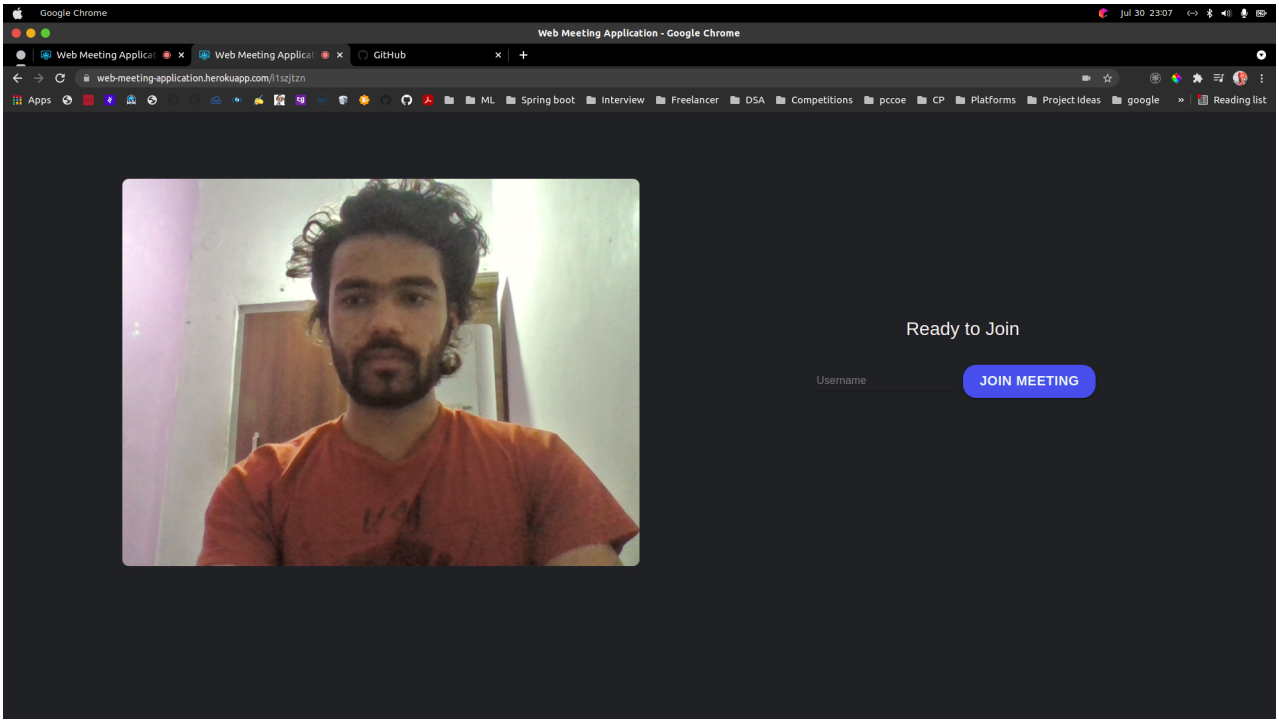
```
   margin-right: 5%;
   margin-top: 5%;
}

.screen_mic:hover, .screen_video:hover{
   cursor: pointer;
   background-color: #8a8c8e;
}

.screen_mic_red:hover, .screen_video_red:hover{
   cursor: pointer;
   background-color: #ee685d;
}

.screen_mic_icon, .screen_video_icon{
   margin-top: 25%;
   font-size: 28px;
}

/* end screen icons  */

.chat{
   cursor: pointer;
}

.people{
   cursor: pointer;
}

.pre_window_input{
   color: whitesmoke;
   border-bottom: whitesmoke;
}
```

Output:

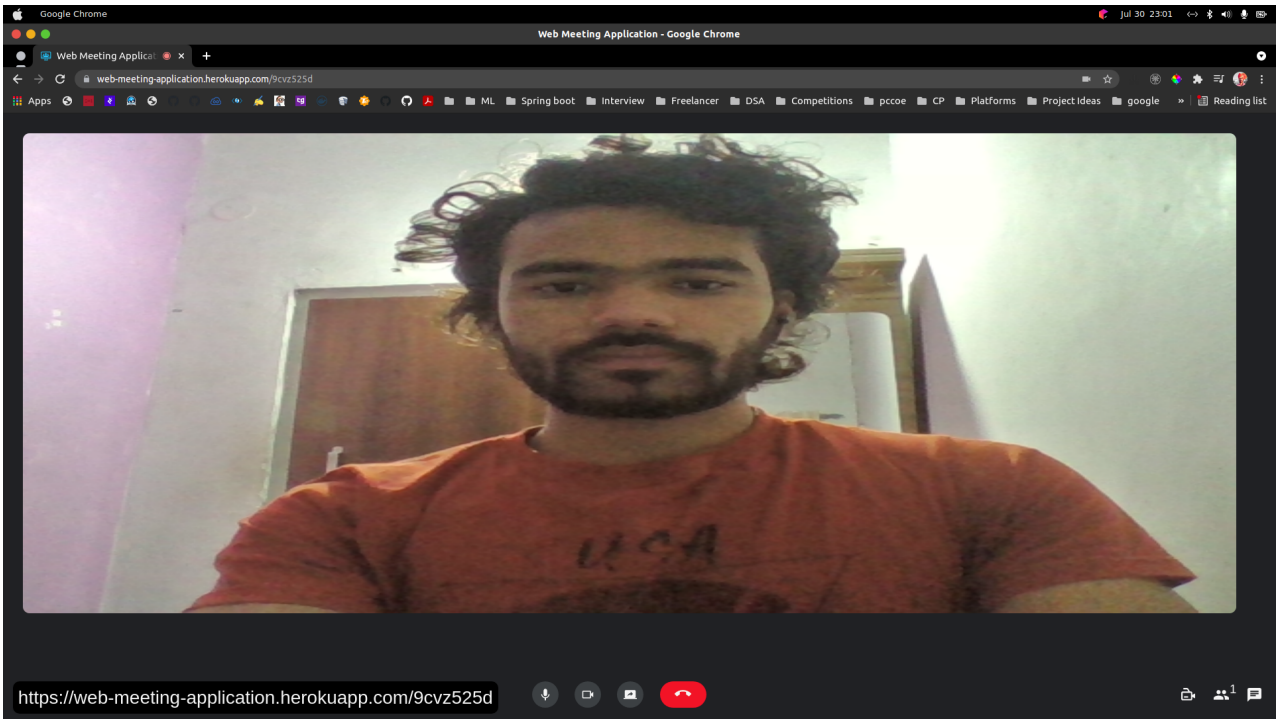
Home Screen :



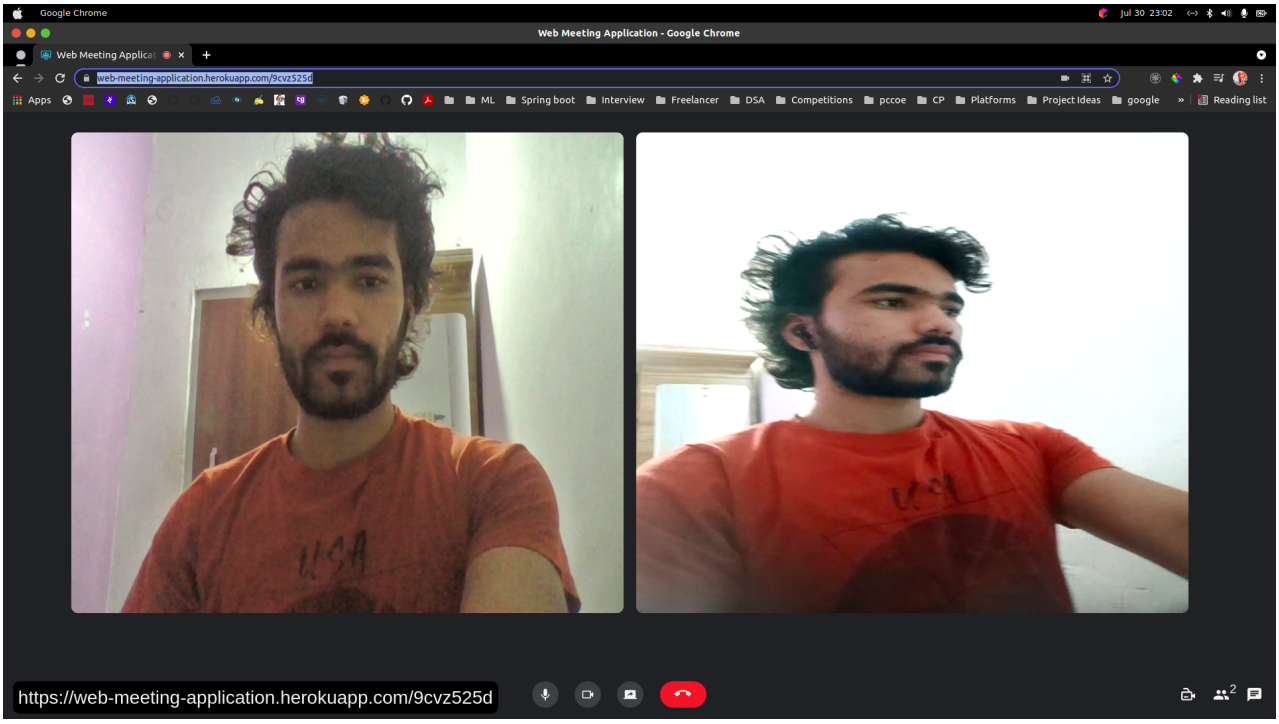
Middle Screen :



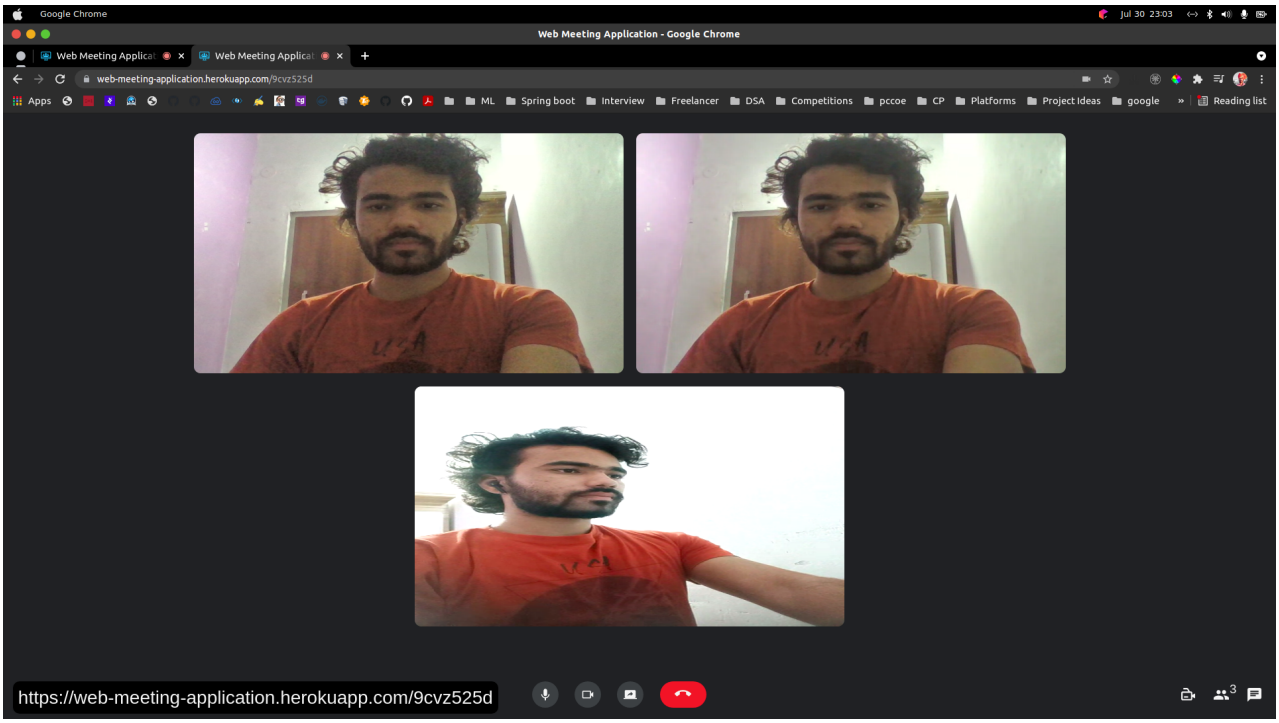
Joined :



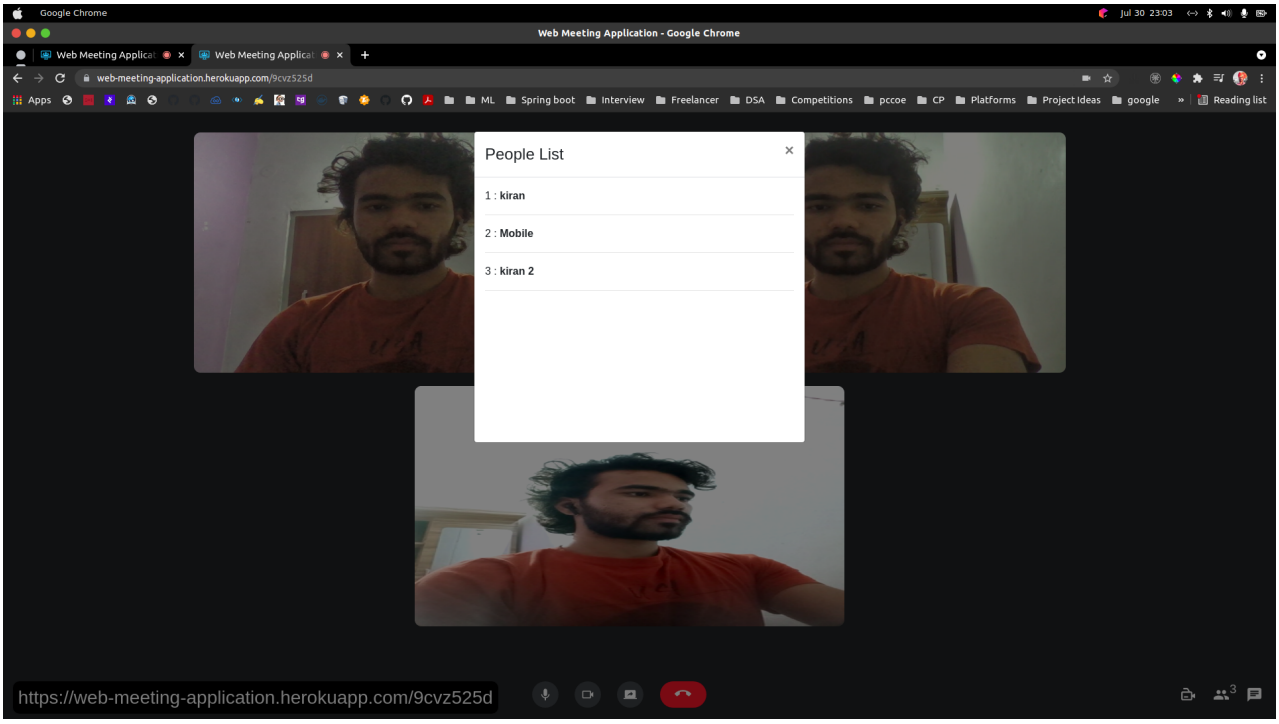
2 People joined :

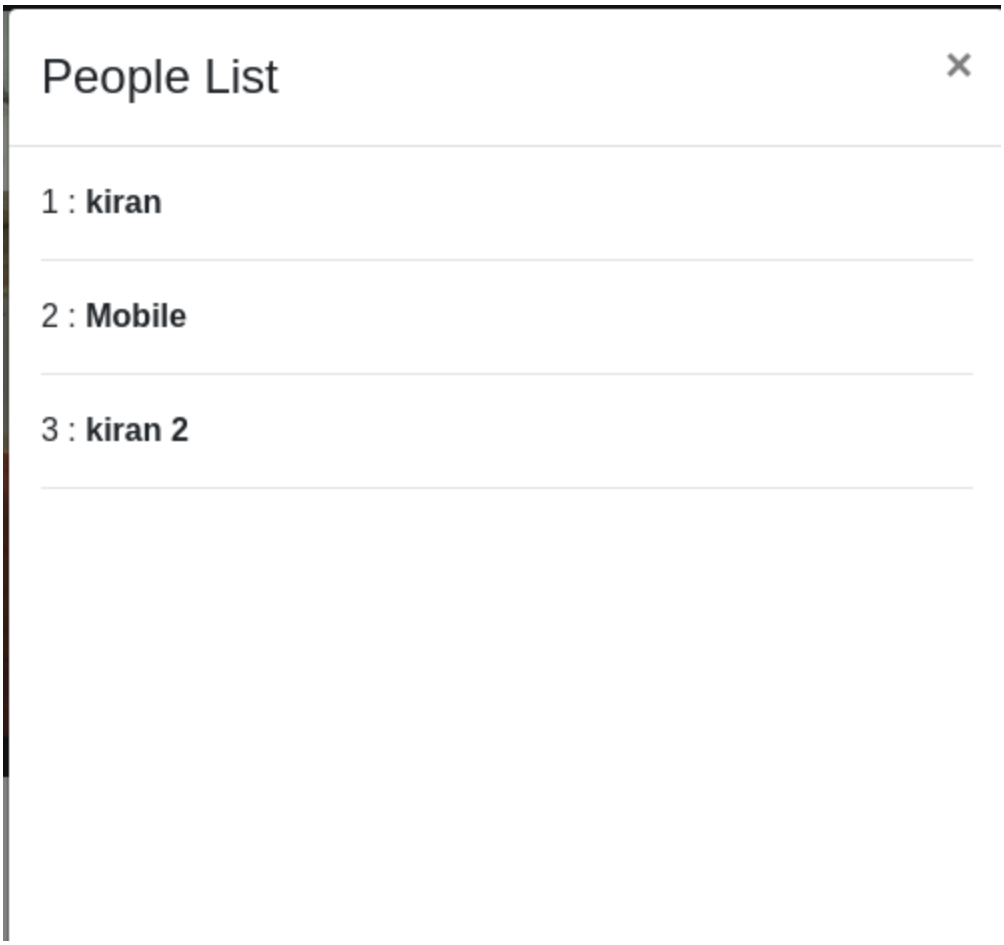


3 Pople joined with dynamic css with js:

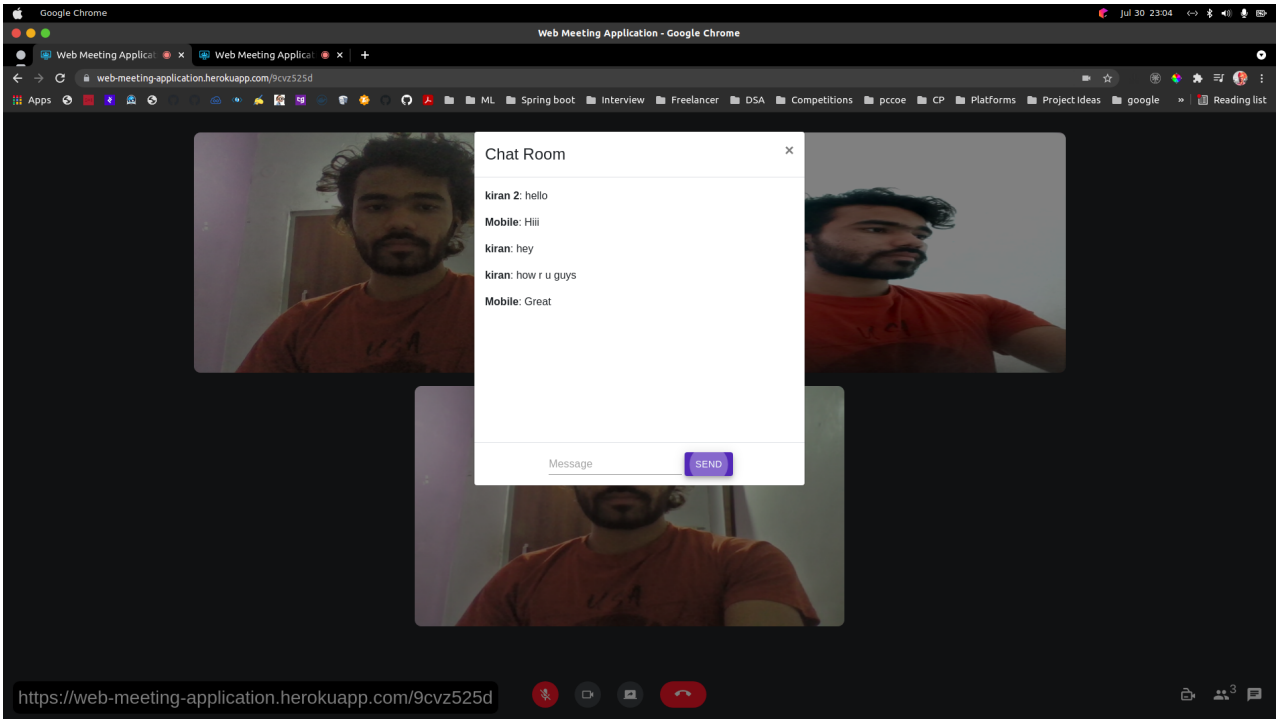


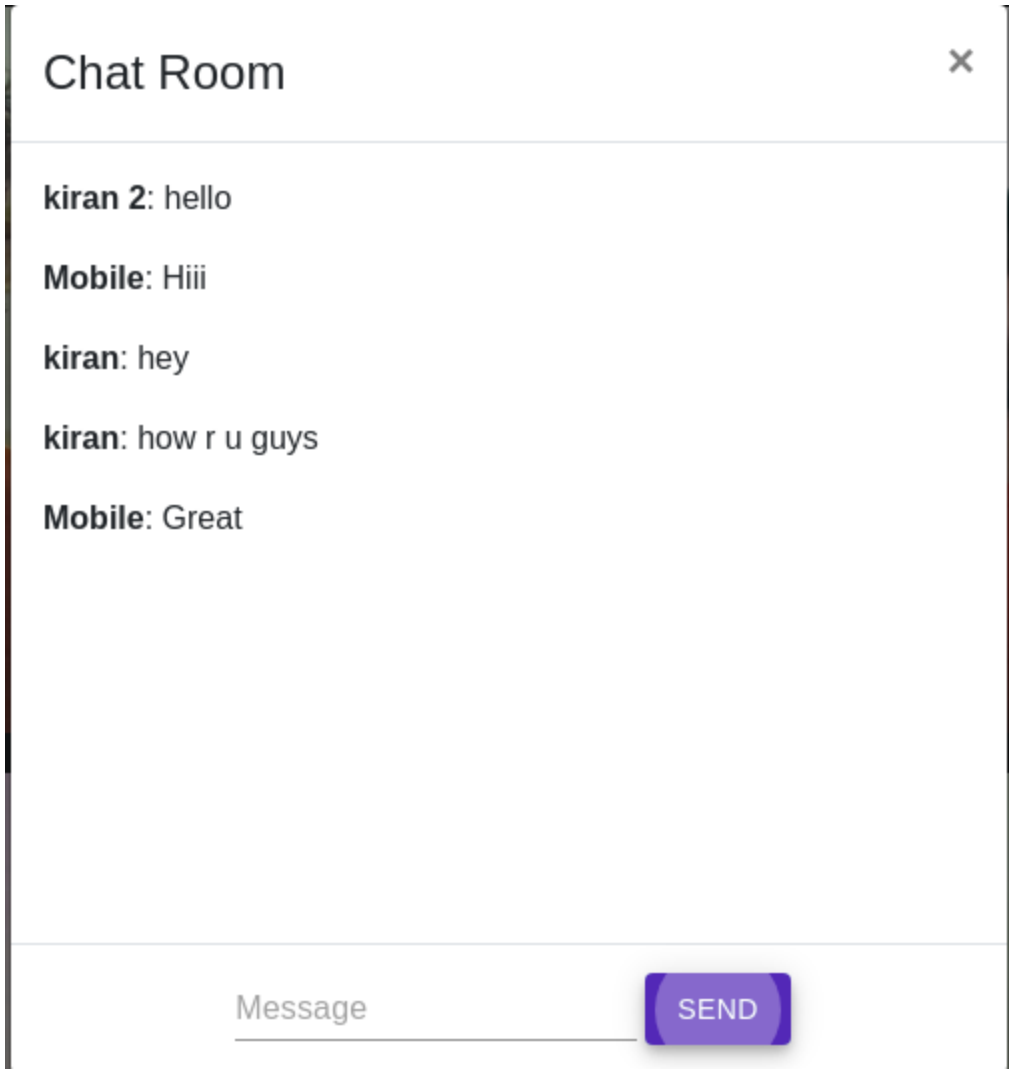
Peoples List :



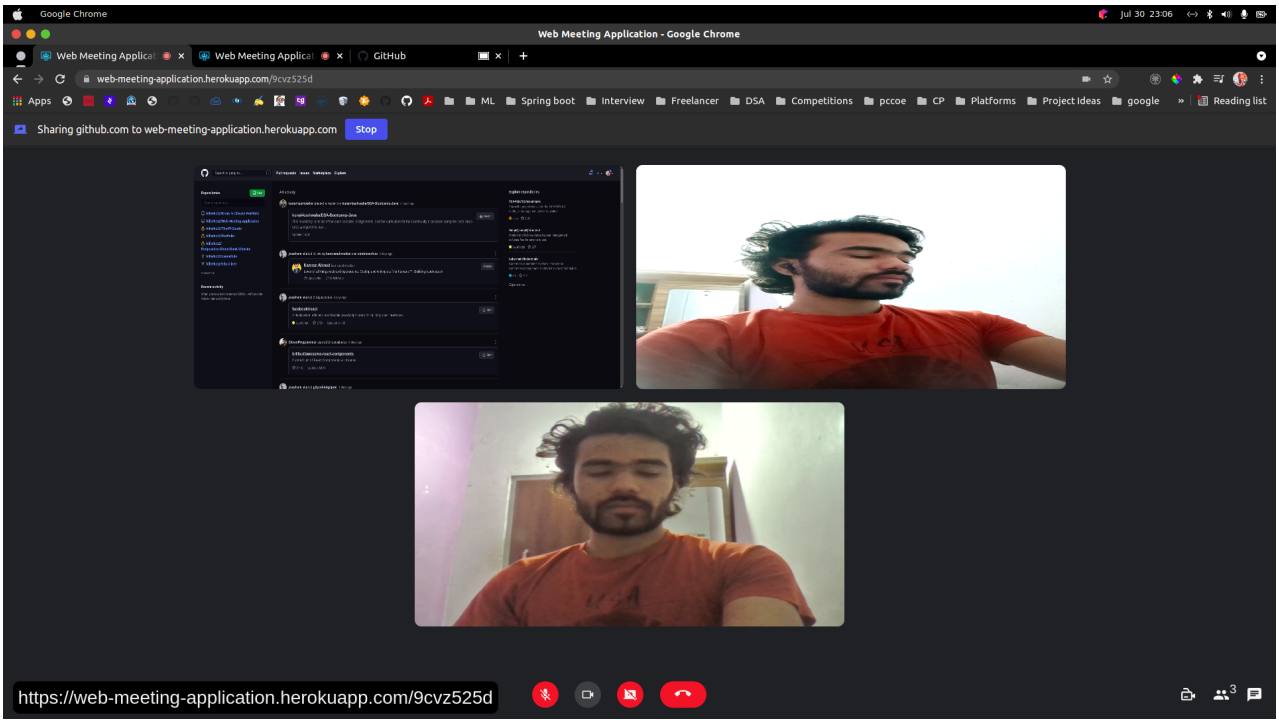

People List
1 : kiran
2 : Mobile
3 : kiran 2

Chat Room :

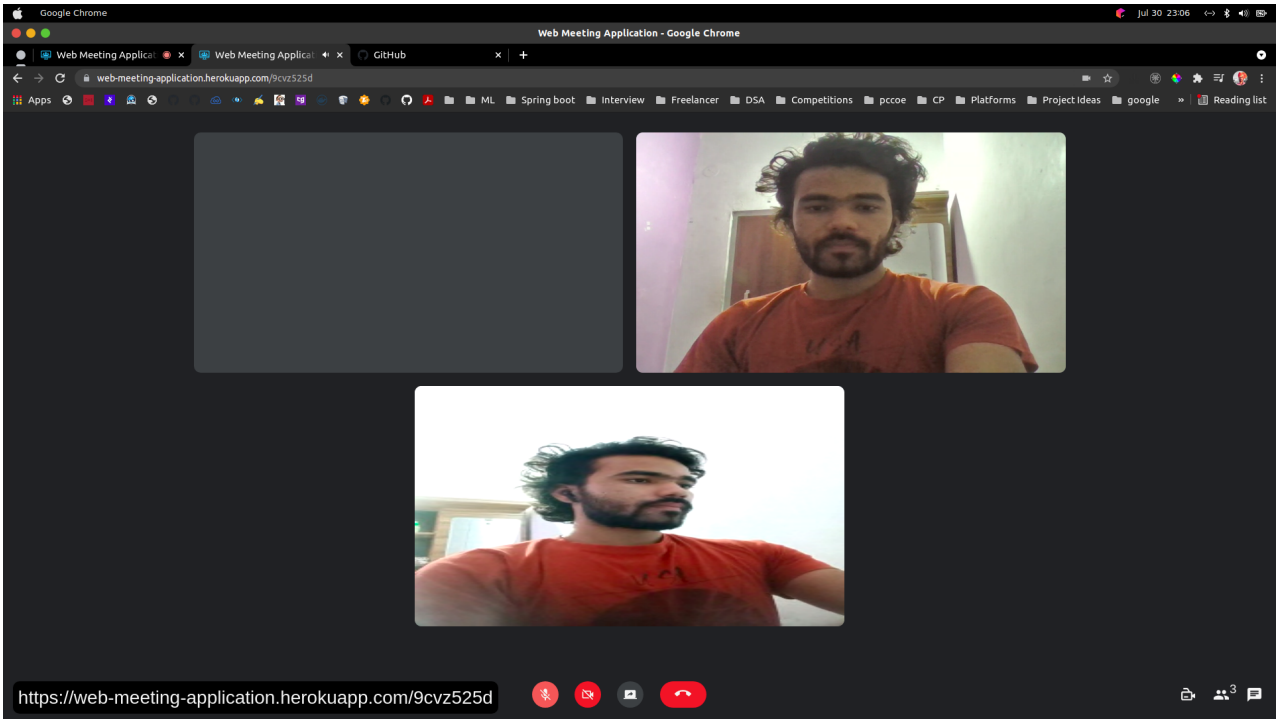

Google Chrome
Web Meeting Application - Google Chrome
Web Meeting Applicat
Web Meeting Applicat
web-meeting-application.herokuapp.com/9cvz525d
Apps
ML
Spring boot
Interview
Freelancer
DSA
Competitions
pccoe
CP
Platforms
Project Ideas
google
Reading list
Chat Room
kiran 2: hello
Mobile: Hiii
kiran: hey
kiran: how r u guys
Mobile: Great
Message
SEND
https://web-meeting-application.herokuapp.com/9cvz525d

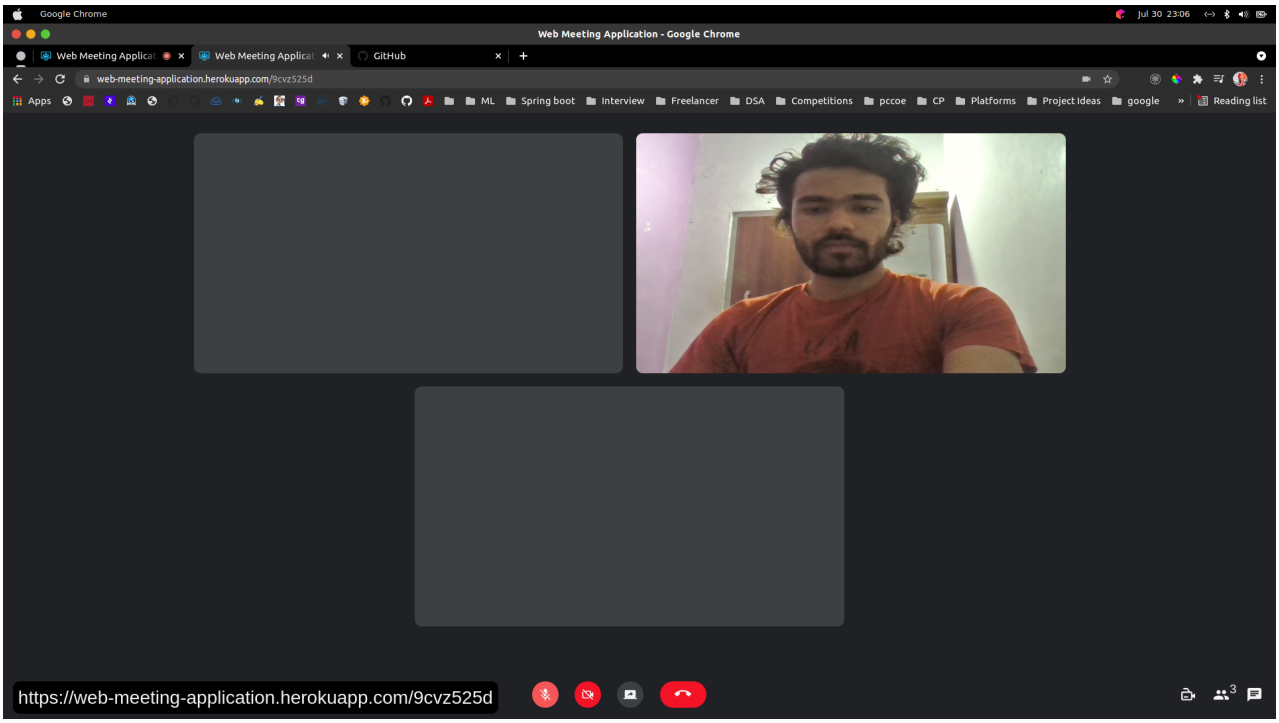


To Share the Screen :

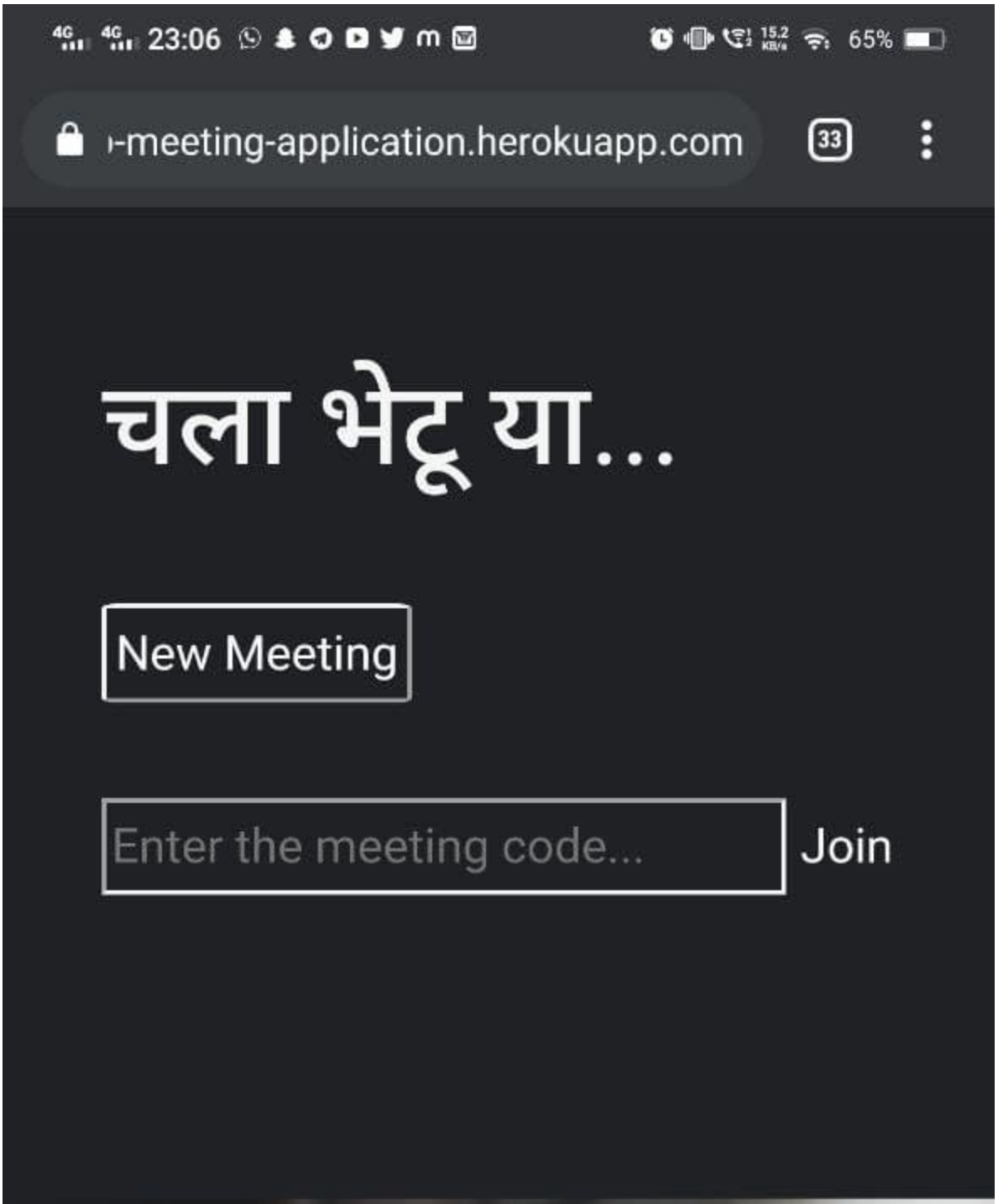


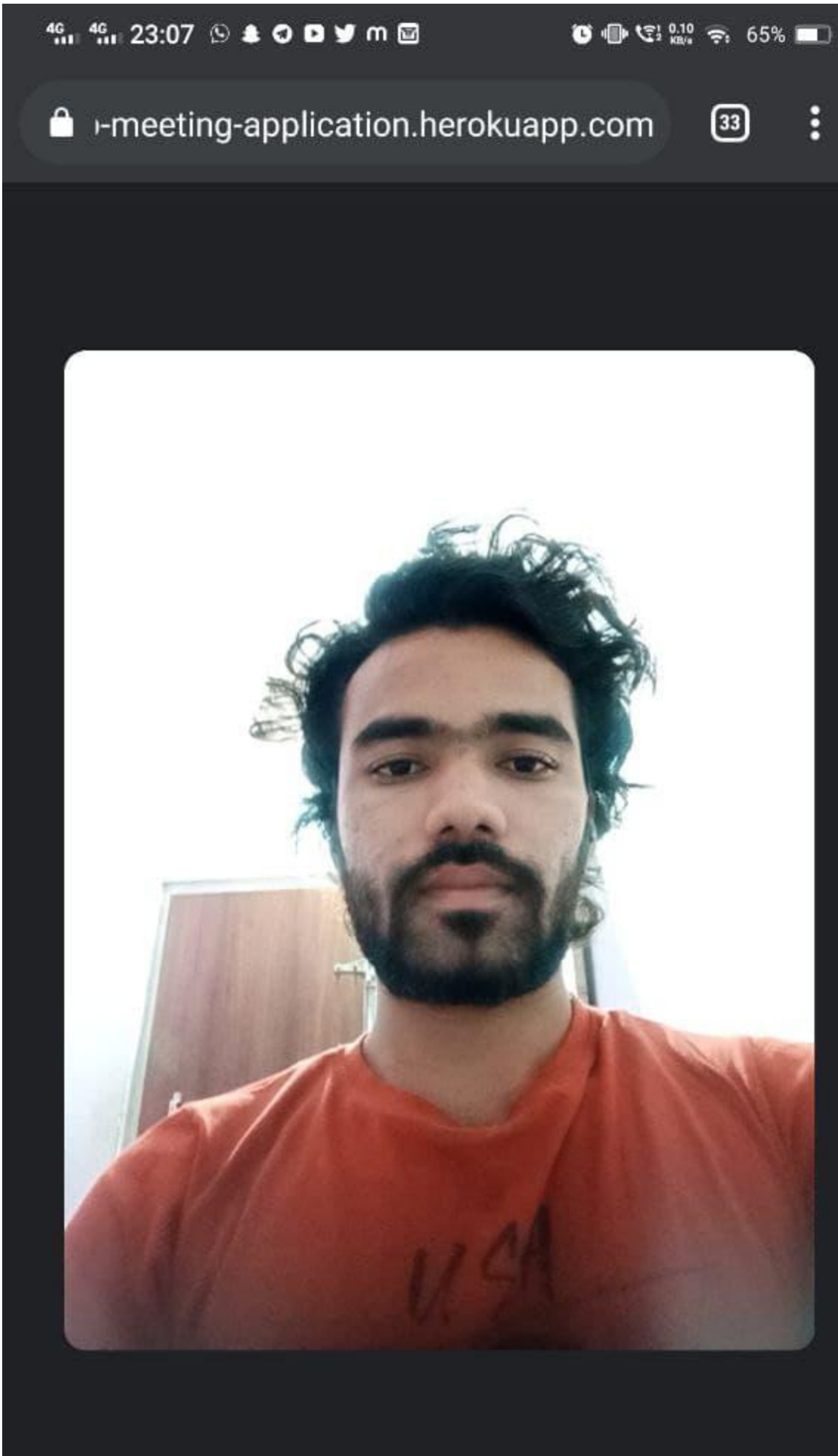
Disable audio and Video :




Google Chrome
Jul 30 23:06
Web Meeting Application - Google Chrome
Web Meeting Applicat
Web Meeting Applicat
GitHub
web-meeting-application.herokuapp.com/9cvz525d
Apps
ML
Spring boot
Interview
Freelancer
DSA
Competitions
pccoe
CP
Platforms
Project Ideas
google
Reading list
https://web-meeting-application.herokuapp.com/9cvz525d

Mobile View :


23:06
65%
-meeting-application.herokuapp.com
33
चला भेटू या...
New Meeting
Enter the meeting code...
Join


23:07
65%
-meeting-application.herokuapp.com
33
Ready to Join


23:02
68%
-meeting-application.herokuapp.com
33


23:03
67%
-meeting-application.herokuapp.com
33