CODE

HTML(INDEX.html)

```
<html>


<head>
 <style>
body {
  background-color: lightgrey;
  color: blue;
}


h1 {
  background-color: black;
  color: white;
}
</style>
    <title>Video Chat </title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/
css/bootstrap.min.css">
    <link href="main.css" rel="stylesheet">
</head>


<body>
```

```
<h1 style="text-align:center">V-chat</h1><p style="background-
color:white;">send the url to ur friends</p>


    <div class="container-fluid">

        <div class="row h-10 w-80">

            <div class="col">

                <select id="filter" class="w-100 bg-dark text-light ml-2 mt-2 select font-
weight-bold border">

                    <option value="none">Normal</option>

                    <option value="grayscale(100%)">B & W</option>

                    <option value="sepia(100%)">Old School</option>

                    <option value="contrast(150%)">Lumination</option>

                    <option value="blur(20px)">Hidden Mist</option>

                    <option value="invert(100%)">Dracula</option>

                    <option value="hue-rotate(180deg">X-men Beast</option>

                    <option value="saturate(25)">Super Saiyan God</option>

                </select>

            </div>

            <div class="col">

                <div class="float-right mt-3">

                    <input class="form-check-input" type="checkbox" id="theme">

                    <label class="form-check-label" for="theme">

                        Dark Theme

                    </label>

                </div>

            </div>
        </div>
        <div class="row h-90 w-100">
            <div class="col-12 col-sm-6 d-flex justify-content-center">
                <div class="embed-responsive embed-responsive-16by9">
                    <video class="embed-responsive-item" muted></video>
                </div>
            </div>
            <div class="col-10 col-sm-6 d-flex justify-content-center">
                <div id="peerDiv" class="embed-responsive embed-responsive-16by9">
                </div>
            </div>
        </div>
    </div>

    <script src="/socket.io/socket.io.js"></script>
    <script src="bundle.js"></script>
</body></html>
```

CSS CODE:

```
video {
    transform: rotateY(180deg);
}

.select {
    height: 30px;
}

.h-90 {
    height: 90% !important;
}

.h-10 {
    height: 10% !important;
}

#peerVideo:hover {
    opacity: 0.6;
    cursor: pointer;
}

.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
```

<u>JAVASCRIPT(MAIN.JS)</u>

```
let Peer = require('simple-peer')
let socket = io()
const video = document.querySelector('video')
const filter = document.querySelector('#filter')
const checkboxTheme = document.querySelector('#theme')
let client = {}
let currentFilter
```

```
//get stream
navigator.mediaDevices.getUserMedia({ video: true, audio: true })
    .then(stream => {
        socket.emit('NewClient')
        video.srcObject = stream
        video.play()

        filter.addEventListener('change', (event) => {
            currentFilter = event.target.value
            video.style.filter = currentFilter
            SendFilter(currentFilter)
   event.preventDefault
        })

        //used to initialize a peer
        function InitPeer(type) {
            let peer = new Peer({ initiator: (type == 'init') ? true : false, stream: stream,
trickle: false })
            peer.on('stream', function (stream) {
                CreateVideo(stream)
            })
            //This isn't working in chrome; works perfectly in firefox.
            // peer.on('close', function () {
            //     document.getElementById("peerVideo").remove();
            //     peer.destroy()
            // })
            peer.on('data', function (data) {
                let decodedData = new TextDecoder('utf-8').decode(data)
                let peervideo = document.querySelector('#peerVideo')
                peervideo.style.filter = decodedData
            })
            return peer
        }

        //for peer of type init
        function MakePeer() {
            client.gotAnswer = false
            let peer = InitPeer('init')
            peer.on('signal', function (data) {
                if (!client.gotAnswer) {
                    socket.emit('Offer', data)
                }
            })
            client.peer = peer
        }

        //for peer of type not init
        function FrontAnswer(offer) {
            let peer = InitPeer('notInit')
            peer.on('signal', (data) => {
                socket.emit('Answer', data)
            })
            peer.signal(offer)
            client.peer = peer
        }

        function SignalAnswer(answer) {
            client.gotAnswer = true
            let peer = client.peer
            peer.signal(answer)
        }

        function CreateVideo(stream) {
            CreateDiv()

            let video = document.createElement('video')
            video.id = 'peerVideo'
            video.srcObject = stream
            video.setAttribute('class', 'embed-responsive-item')
            document.querySelector('#peerDiv').appendChild(video)
            video.play()
            //wait for 1 sec
            setTimeout(() => SendFilter(currentFilter), 1000)

            video.addEventListener('click', () => {
                if (video.volume != 0)
```

```
            video.volume = 0
        else
            video.volume = 1
    })

}

function SessionActive() {
    document.write('Session Active. Please come back later')
}

function SendFilter(filter) {
    if (client.peer) {
        client.peer.send(filter)
    }
}

function RemovePeer() {
    document.getElementById("peerVideo").remove();
    document.getElementById("muteText").remove();
    if (client.peer) {
        client.peer.destroy()
    }
}
```

```
        socket.on('BackOffer', FrontAnswer)
        socket.on('BackAnswer', SignalAnswer)
        socket.on('SessionActive', SessionActive)
        socket.on('CreatePeer', MakePeer)
        socket.on('Disconnect', RemovePeer)

    })
    .catch(err => document.write(err))

checkboxTheme.addEventListener('click', () => {
    if (checkboxTheme.checked == true) {
        document.body.style.backgroundColor = '#212529'
        if (document.querySelector('#muteText'))
  document.querySelector('#muteText').style.color = "#fff"
        }


    }
    else {
        document.body.style.backgroundColor = '#fff'
        if (document.querySelector('#muteText')) {
            document.querySelector('#muteText').style.color = "#212529"
        }
    }
}
)
```

```
function CreateDiv() {
    let div = document.createElement('div')
    div.setAttribute('class', "centered")
    div.id = "muteText"
    div.innerHTML = "Click to Mute/Unmute"
    document.querySelector('#peerDiv').appendChild(div)
    if (checkboxTheme.checked == true)
document.querySelector('#muteText').style.color = "#ffff"}
```

<u>JAVASCRIPT(SERVER.JS)</u>

```
const express = require('express')
const app = express()
const http = require('http').Server(app)
const io = require('socket.io')(http)
const port = process.env.PORT || 3000

app.use(express.static(__dirname,public)),
app.listen(process.env.PORT);
let clients = 0

io.on('connection', function (socket) {
    socket.on("NewClient", function () {
        if (clients < 2) {
            if (clients == 1) {
                this.emit('CreatePeer')
            }
        }
        else
            this.emit('SessionActive')
        clients++;
    })
    socket.on('Offer', SendOffer)
    socket.on('Answer', SendAnswer)
    socket.on('disconnect', Disconnect)
})

function Disconnect() {
    if (clients > 0) {
        if (clients <= 2)
            this.broadcast.emit("Disconnect")
        clients--
    }
}

function SendOffer(offer) {
    this.broadcast.emit("BackOffer", offer)
}
```

```
function SendAnswer(data) {
    this.broadcast.emit("BackAnswer", data)
}

http.listen(port, () => console.log(`Active on ${port} port`))
```