TASK 7

8. VoIP: Incorporate a feature into the web app that allows users to make video calls to their friends and share screen of youtube website and we should have option to record the video session and save it in local,

Step 1: Set Up Your Project in VS Code

Open VS Code and create a new folder for your project.

Inside the folder, create the following files:

index.html

Step 2: Write the HTML Structure

Create and open the index.html file, then add the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>VoIP Video Call with Screen Sharing and Recording</title>
  <style>
    body, html {
      margin: 0;
      padding: 0;
      height: 100%;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      background-color: #f0f0f0;
    }

    #video-container {
      display: flex;
      justify-content: center;
      gap: 20px;
      margin-top: 20px;
    }
```

```
    video {
      width: 300px;
      height: 200px;
      background-color: black;
    }

    #screen-share-button, #record-button {
      margin: 10px;
      padding: 10px 20px;
      background-color: #007bff;
      color: white;
      border: none;
      cursor: pointer;
    }

    #recording-status {
      margin-top: 10px;
      color: red;
    }
  </style>
</head>
```

```
<body>

<h1>Video Call with Screen Sharing and Recording</h1>

<button id="screen-share-button">Share Screen</button>
<button id="record-button">Start Recording</button>

<div id="video-container">
  <video id="local-video" autoplay muted></video>
  <video id="remote-video" autoplay></video>
</div>

<div id="recording-status"></div>

<script>
  const screenShareButton =
document.getElementById('screen-share-button');
  const recordButton = document.getElementById('record-
button');
  const localVideo = document.getElementById('local-
video');
```

```
 const remoteVideo = document.getElementById('remote-video');
 const recordingStatus = document.getElementById('recording-status');

 let localStream;
 let remoteStream;
 let peerConnection;
 let mediaRecorder;
 let recordedChunks = [];
 let isRecording = false;

 // WebRTC configuration
 const peerConfig = {
  iceServers: [
   { urls: 'stun:stun.l.google.com:19302' }
  ]
 };

 // Initialize video call
 async function initCall() {
```

```
  try {
    // Access user's camera and microphone
    localStream = await navigator.mediaDevices.getUserMedia({ video: true, audio: true });
    localVideo.srcObject = localStream;

    // Create peer connection and add local stream
    peerConnection = new RTCPeerConnection(peerConfig);
    localStream.getTracks().forEach(track => peerConnection.addTrack(track, localStream));

    // Set up ICE candidate handling
    peerConnection.onicecandidate = event => {
     if (event.candidate) {
       // Send the ICE candidate to the other peer (using your signaling mechanism)
       console.log('New ICE candidate', event.candidate);
     }
    };
```

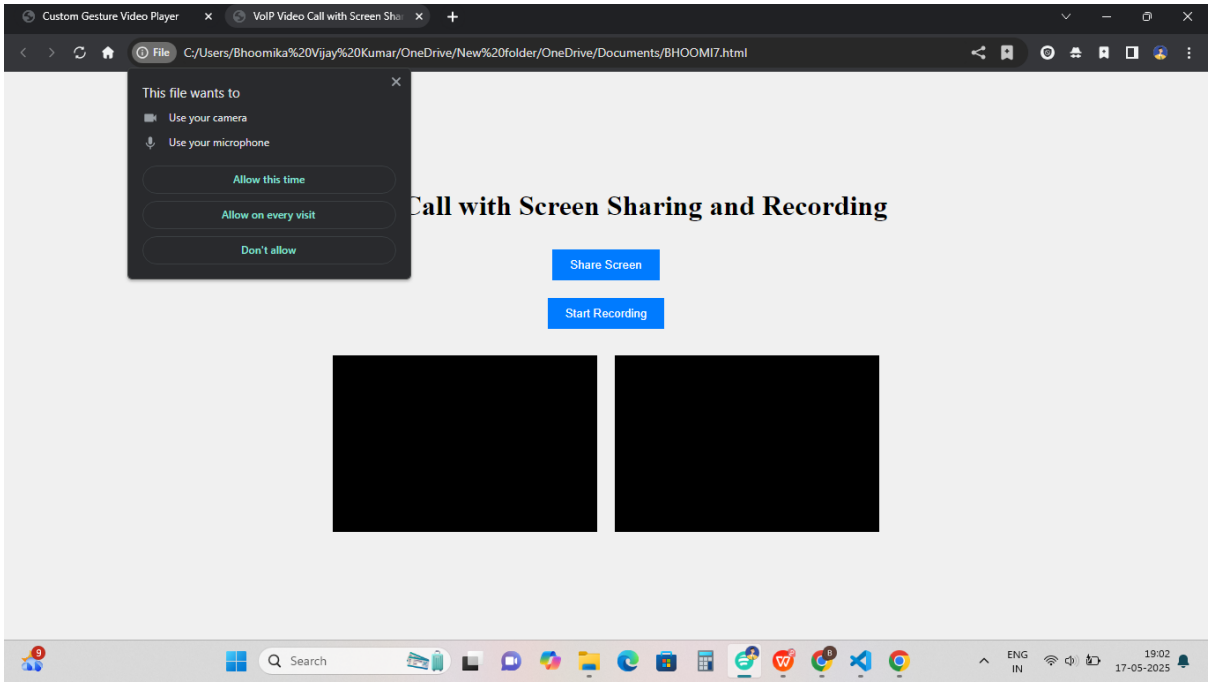
```
    // Handle remote stream
    peerConnection.ontrack = event => {
      if (!remoteStream) {
        remoteStream = event.streams[0];
        remoteVideo.srcObject = remoteStream;
      }
    };
  } catch (err) {
    console.error('Error accessing media devices:', err);
  }
}

// Start screen sharing
async function shareScreen() {
  try {
    const screenStream = await
navigator.mediaDevices.getDisplayMedia({ video: true });
    const screenTrack = screenStream.getTracks()[0];
    const senders = peerConnection.getSenders();
    const screenSender = senders.find(sender =>
sender.track.kind === 'video');
```

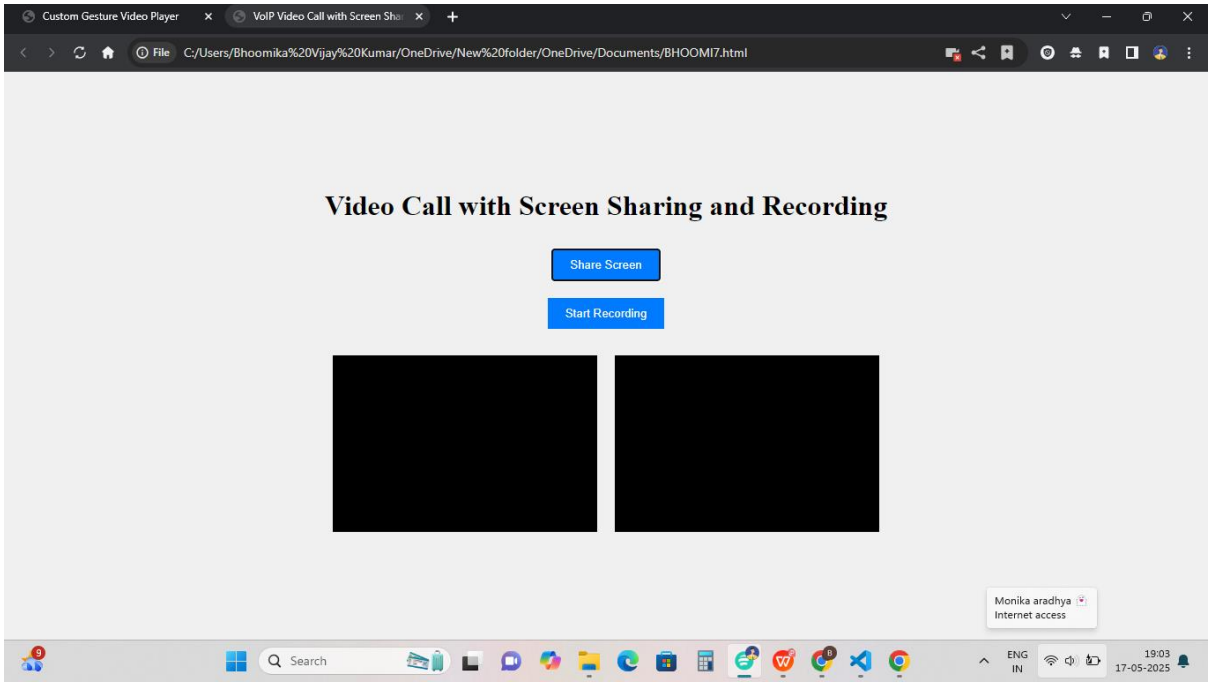
```
    // Replace video track with screen track
    if (screenSender) {
      screenSender.replaceTrack(screenTrack);
    }

    // Show the screen in local video element
    localVideo.srcObject = screenStream;

    // Update UI
    screenShareButton.disabled = true;
  } catch (err) {
    console.error('Error sharing screen:', err);
  }
}

// Handle recording
function startRecording() {
  if (!isRecording) {
    recordedChunks = [];
    mediaRecorder = new MediaRecorder(localStream);
```

```
    mediaRecorder.ondataavailable = event => {
      recordedChunks.push(event.data);
    };

    mediaRecorder.onstop = () => {
      const blob = new Blob(recordedChunks, { type:
'video/webm' });
      const url = URL.createObjectURL(blob);
      const a = document.createElement('a');
      a.href = url;
      a.download = 'video-session.webm';
      a.click();
    };

    mediaRecorder.start();
    isRecording = true;
    recordingStatus.textContent = 'Recording...';
    recordButton.textContent = 'Stop Recording';
  } else {
    mediaRecorder.stop();
```

```
    isRecording = false;
    recordingStatus.textContent = '';
    recordButton.textContent = 'Start Recording';
  }
}

// Signaling: Offer to start a call (simplified here)
async function createOffer() {
  try {
    const offer = await peerConnection.createOffer();
    await peerConnection.setLocalDescription(offer);

    // Send offer to the other peer (via your signaling server)
    console.log('Offer:', offer);
  } catch (err) {
    console.error('Error creating offer:', err);
  }
}

// Call initialization
initCall();
```

```
// Event Listeners
screenShareButton.addEventListener('click', shareScreen);
recordButton.addEventListener('click', startRecording);

</script>

</body>
</html>
```




Custom Gesture Video Player
VoIP Video Call with Screen Shar
File
C:/Users/Bhoomika%20Vijay%20Kumar/OneDrive/New%20folder/OneDrive/Documents/BHOOMI7.html
Video Call with Screen Sharing and Recording
Share Screen
Start Recording
Monika aradhya
Internet access
Search
ENG
IN
19:03
17-05-2025