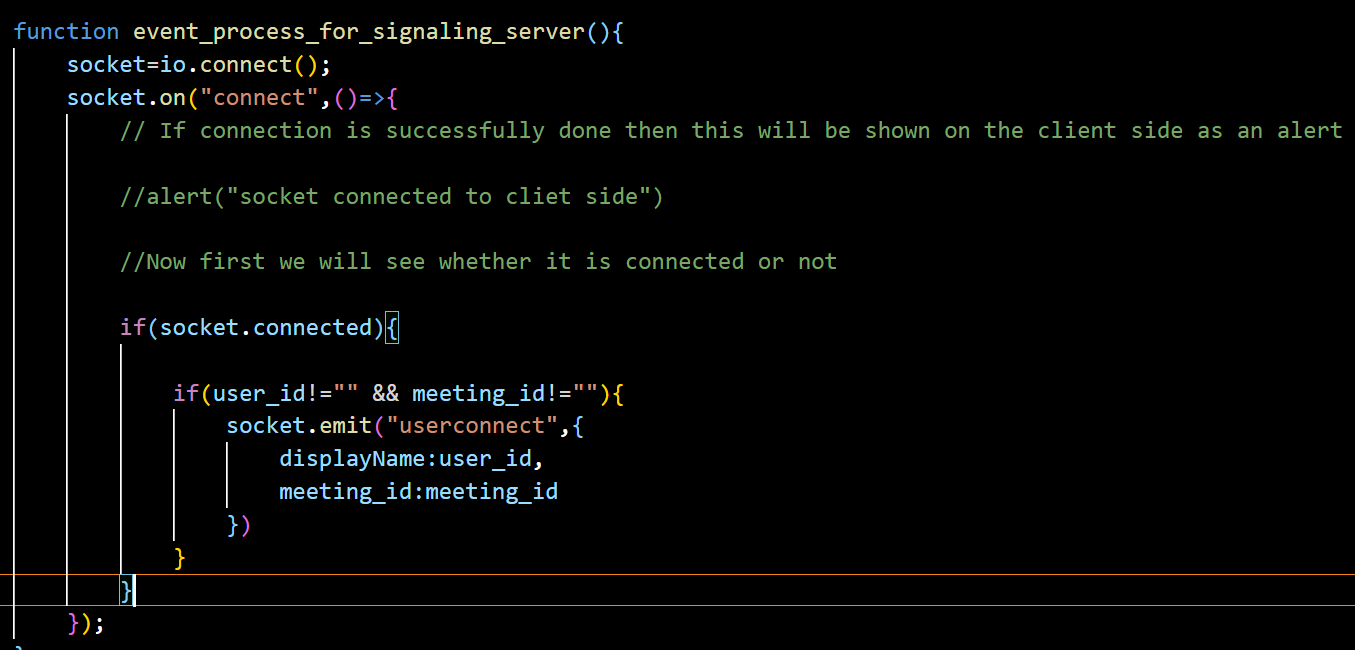
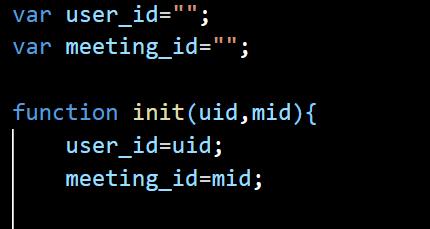
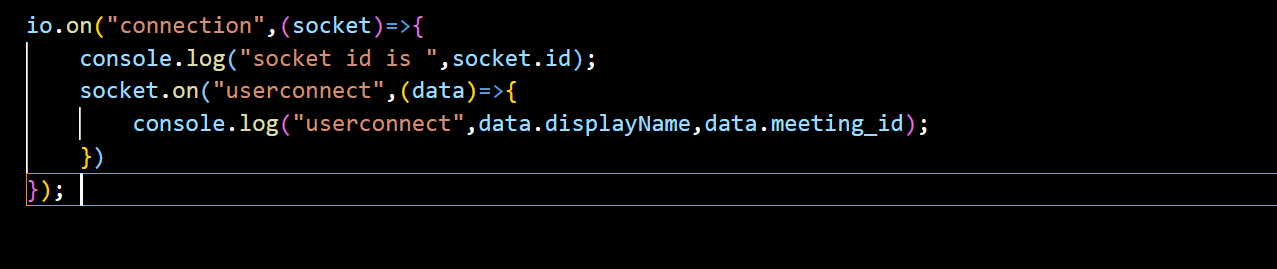
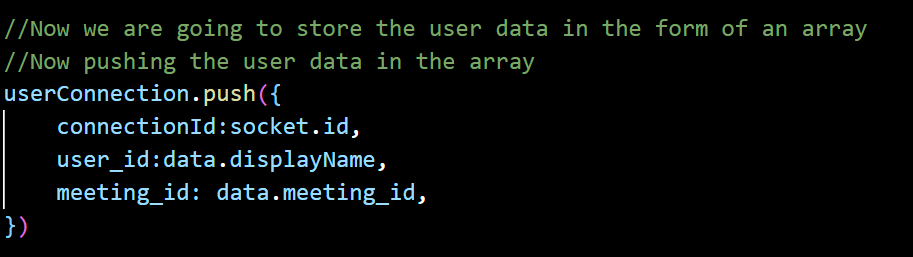
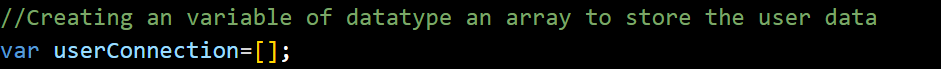
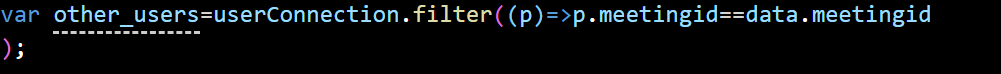
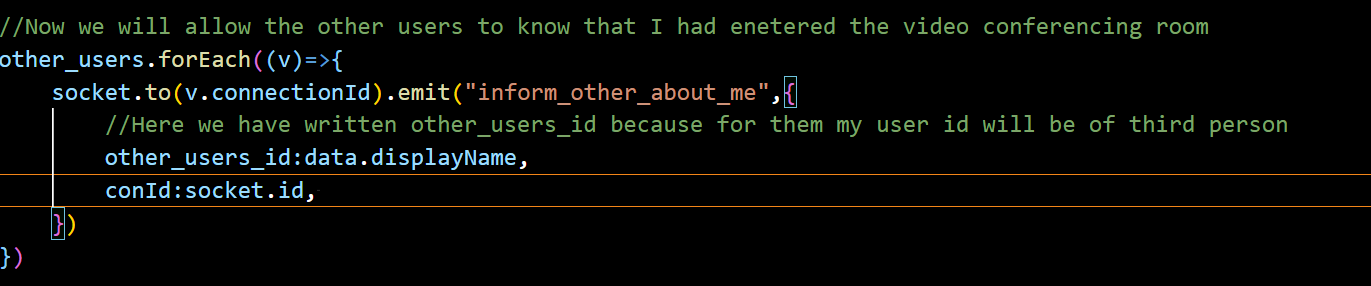
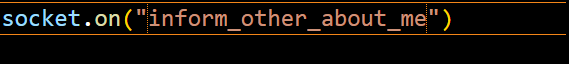
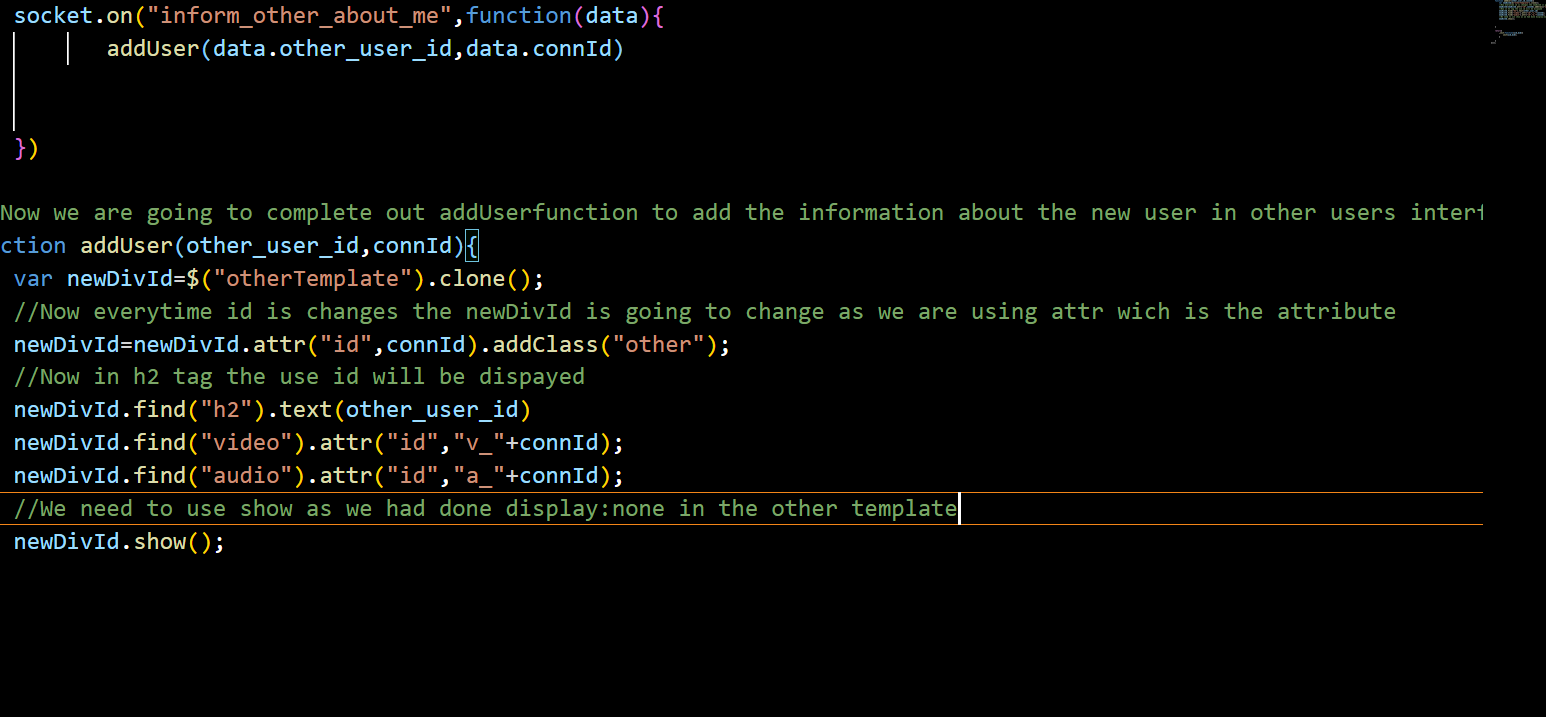
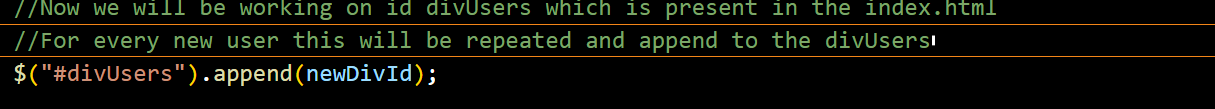
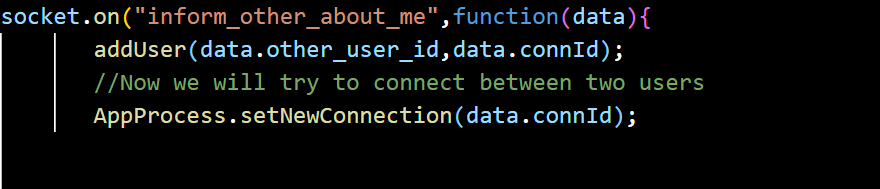
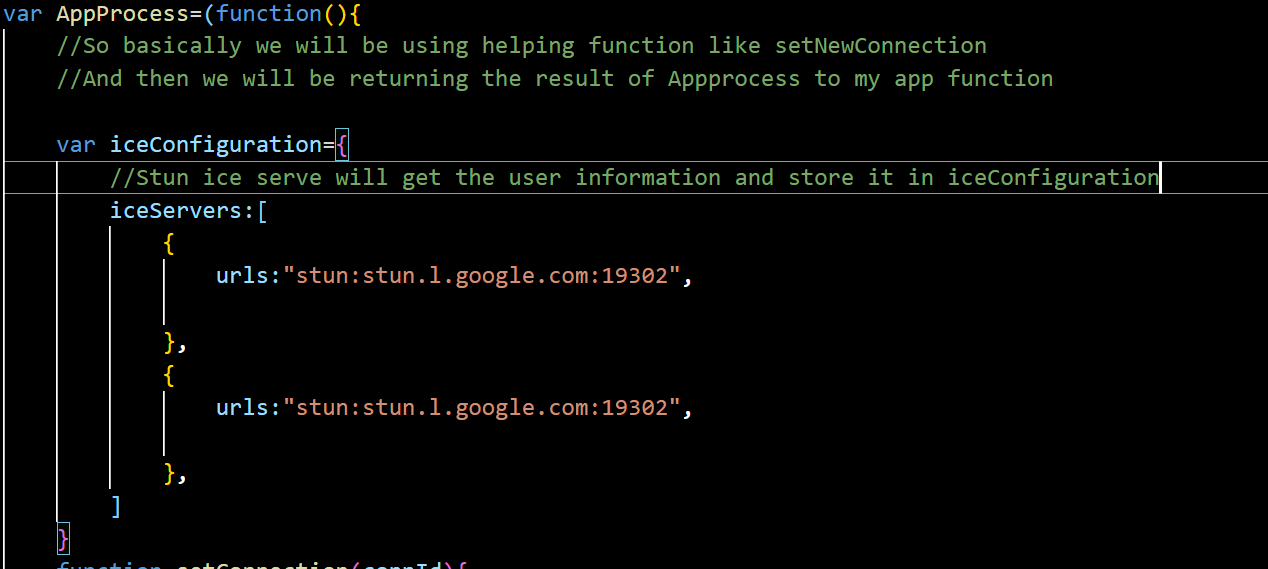
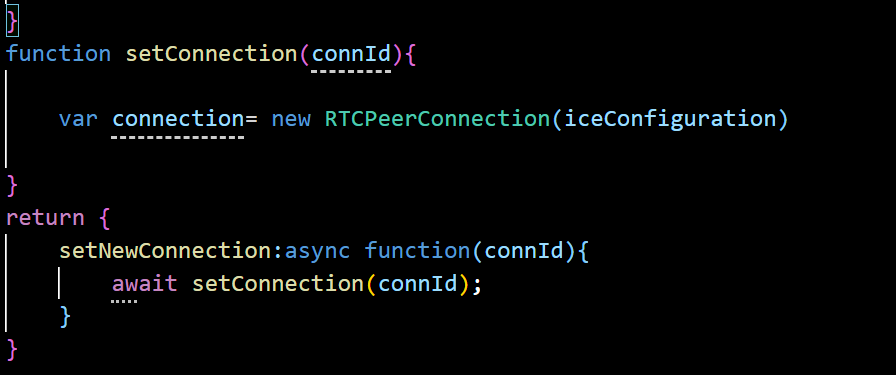
Section 5

* Part1:Let other users know about you
* Now In this we are going to store the user information so that other users can also see it a kind of perfect Group calling system.
* From index.html we got the user id and meeting id and then we sent it to the app.js
* 
* For this we will be declaring the variables in global zone
* 
* Here userConnect is the name of the event
* The code snippet you provided is using the `emit` function to send a custom event called "userconnect" from the `socket` object. Here's a breakdown of what the code is doing:
* 1. The `if(socket.connected)` condition checks if the socket is currently connected to the server. It ensures that the code inside the condition will only be executed if the socket is indeed connected.
* 2. Inside the `if` block, there is another condition `if(user\_id!="" && meeting\_id!="")` which checks if both `user\_id` and `meeting\_id` variables are not empty strings. This condition ensures that the `emit` function is called only if both `user\_id` and `meeting\_id` have valid values.
* 3. If the conditions mentioned above are met, the `socket.emit` function is called with the "userconnect" event name and an object as the second argument. The object contains two properties:
* - `displayName`: The value of the `user\_id` variable.
* - `meeting\_id`: The value of the `meeting\_id` variable.
* By emitting the "userconnect" event with the specified data, you are essentially sending information to the server-side code or any other clients that are listening for this event. The server or other clients can then handle this event and perform the necessary actions based on the received data.
* Now we will send the data to our server side
* 
* 
* 
* 
* in summary, the code filters the **userConnection** array to retrieve only the objects that have a **meetingid** property matching **data.meetingid**. The filtered result is stored in the **other\_users** variable for further processing or usage.
* 
* The **to** function allows you to target a specific socket or a group of sockets based on their identifier. It is often used in scenarios where you want to send an event to a specific client or a subset of clients connected to the server.
* Now in app.js we will call our event
* 

Part-2

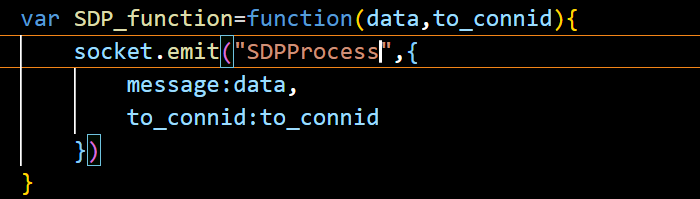
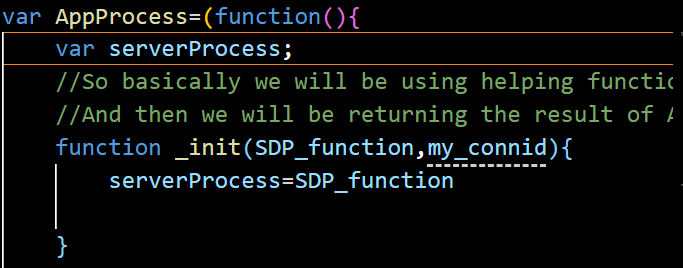
 //Now we are going to complete out addUserfunction to add the information about the new user in other users interface

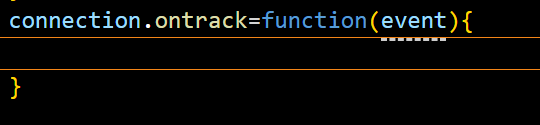
* Now from index.htm we will clone the otherTemplate section first
* 
* 
* Now we run this server then we see that if some user enter the room then also the user which is present in the room does not gets notified so now we need to make a connection between then which can be done by webrtc
* 
* 

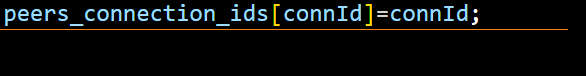
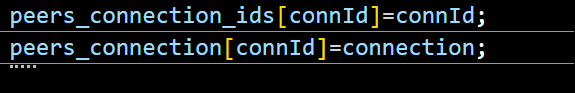
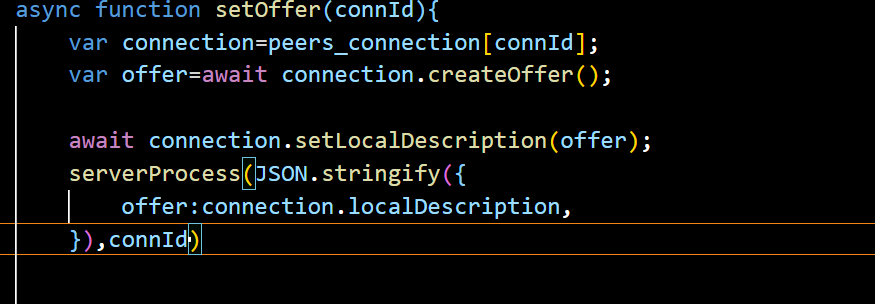


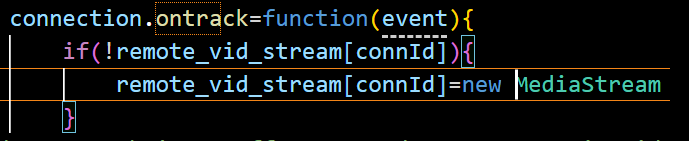
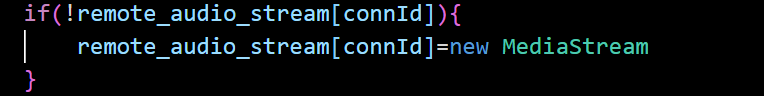
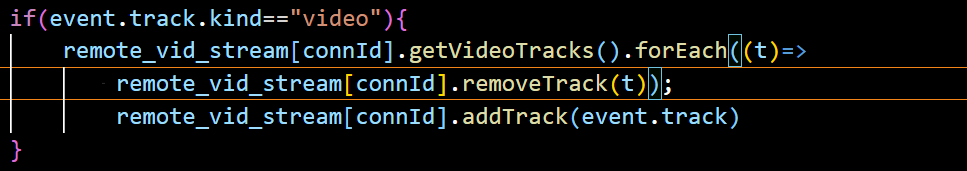
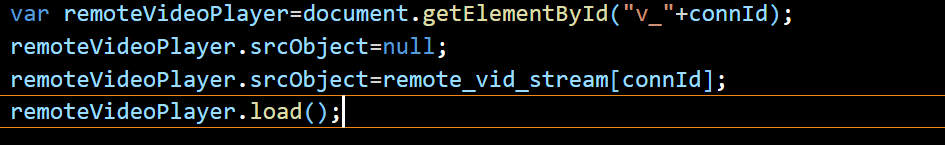
Part 3

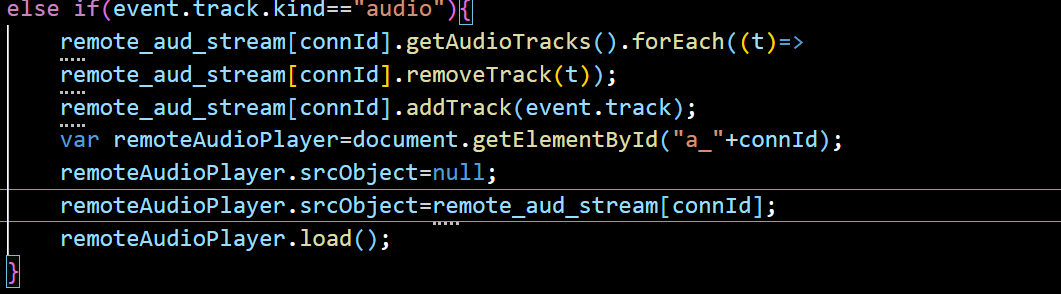
Sending an offer to connect

* Creating an SDP function
* 
* We Will be sending the Sdp function packaged in SDP\_function to \_init(SDPfunction,
* Now we will be connecting it to serverProcess
* So for this first we will declare the server Process
* 
* 

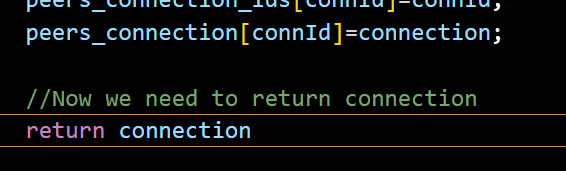


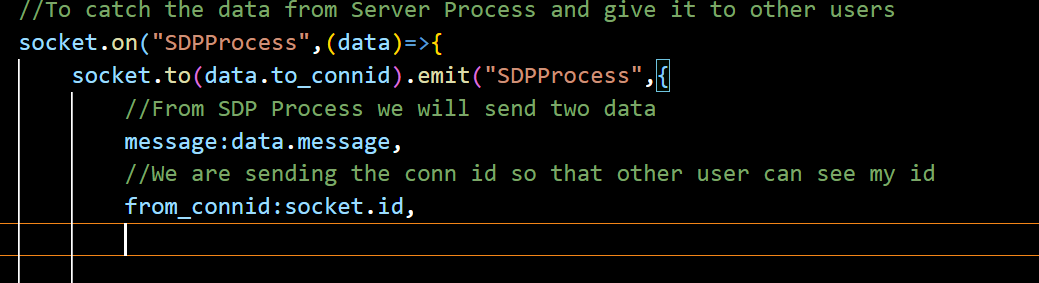
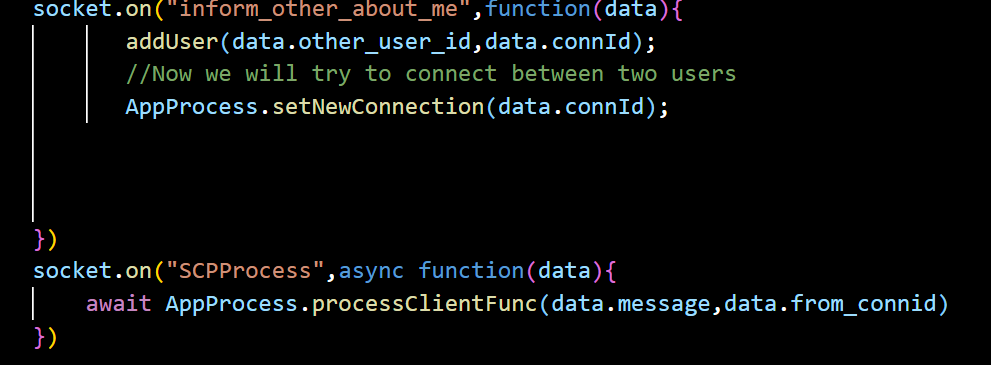
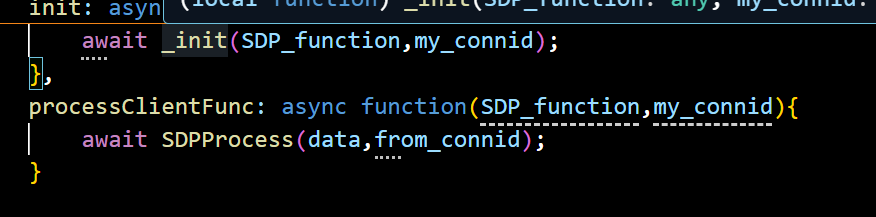
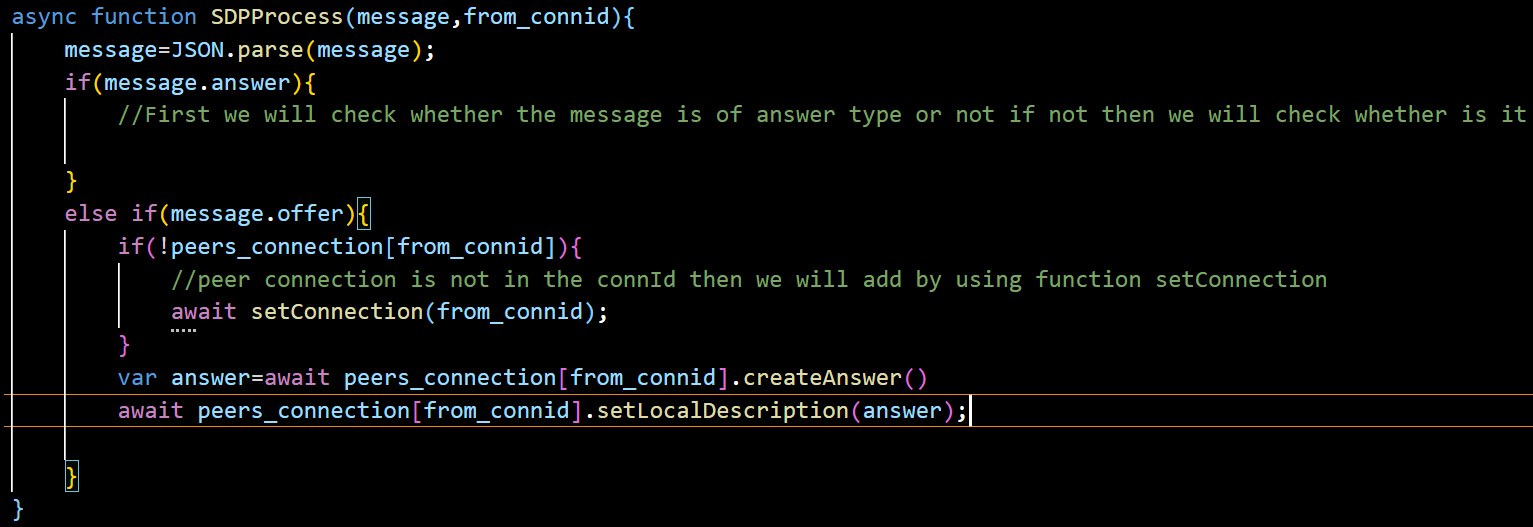
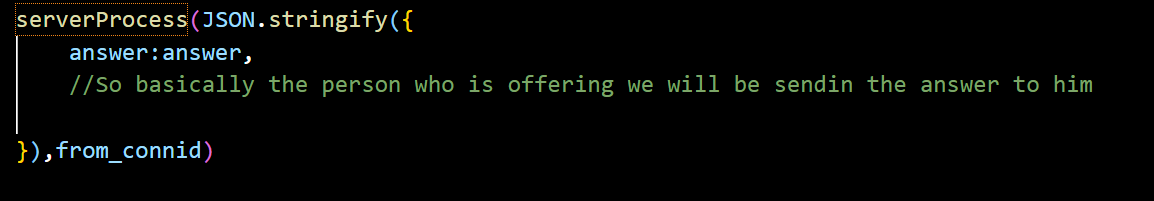
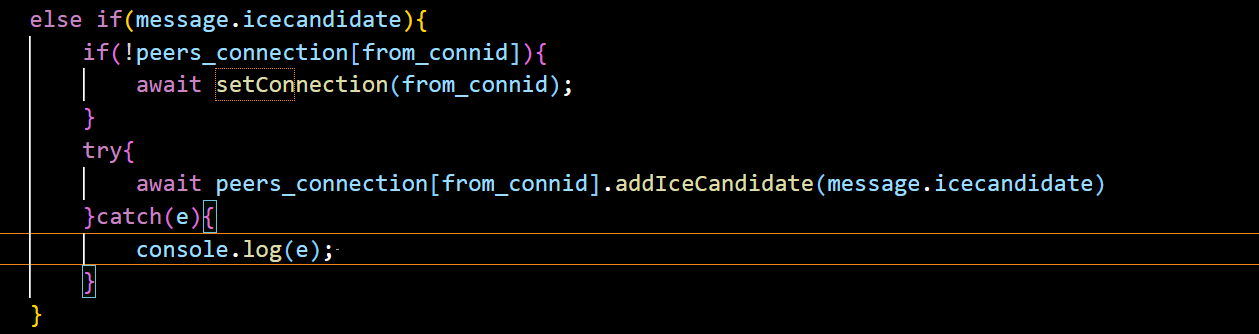
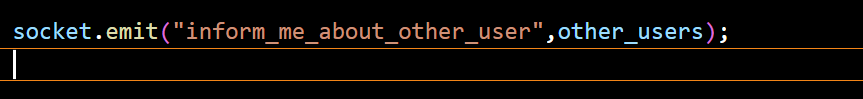
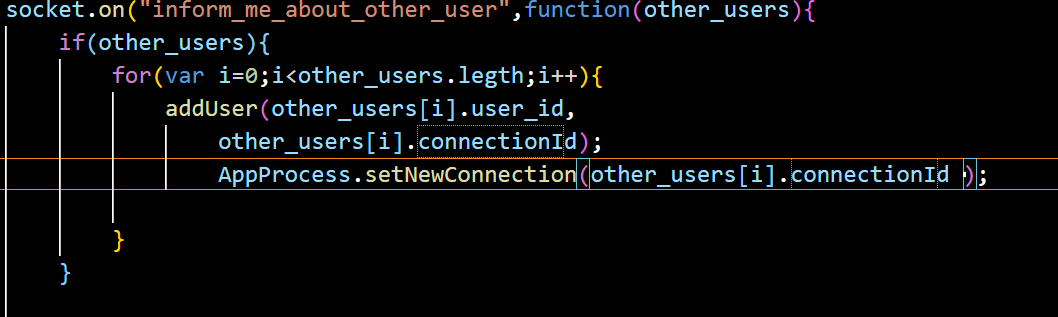
* Now we will be storing connId in peer\_connection in form of an array
* 
* Now we will be storing connection which we got from RTCPeerConnection
* 
* RTCPeerConnection has many offers among them createOffer is one of the property we will be using it
* 

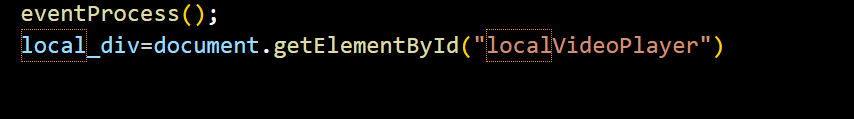
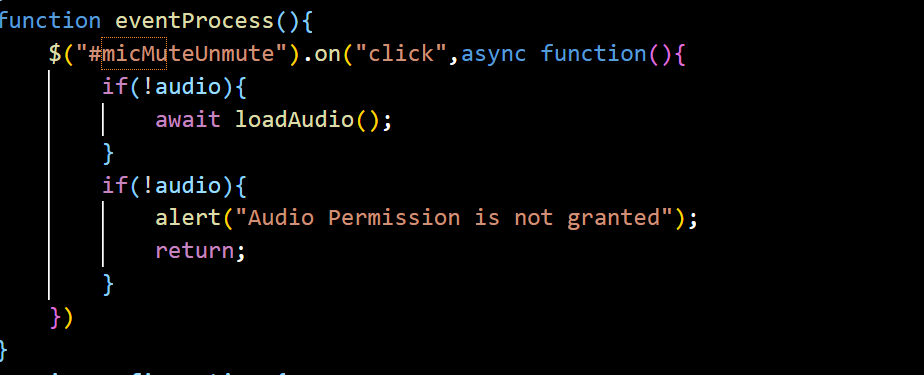
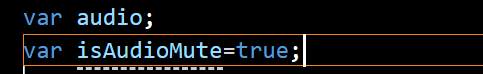
* Completion of set connection function
* 
* Now similar we will do for audio
* 
* Now we will check for whether video Track is set or not if set then we will remove it
* 
* In summary, the code snippet replaces the existing video track in the **remote\_vid\_stream[connId]** media stream object with a new video track when a video track event occurs.
* 
* The above code will run if video track is available on the event or not
* Similarly we will do this for audio

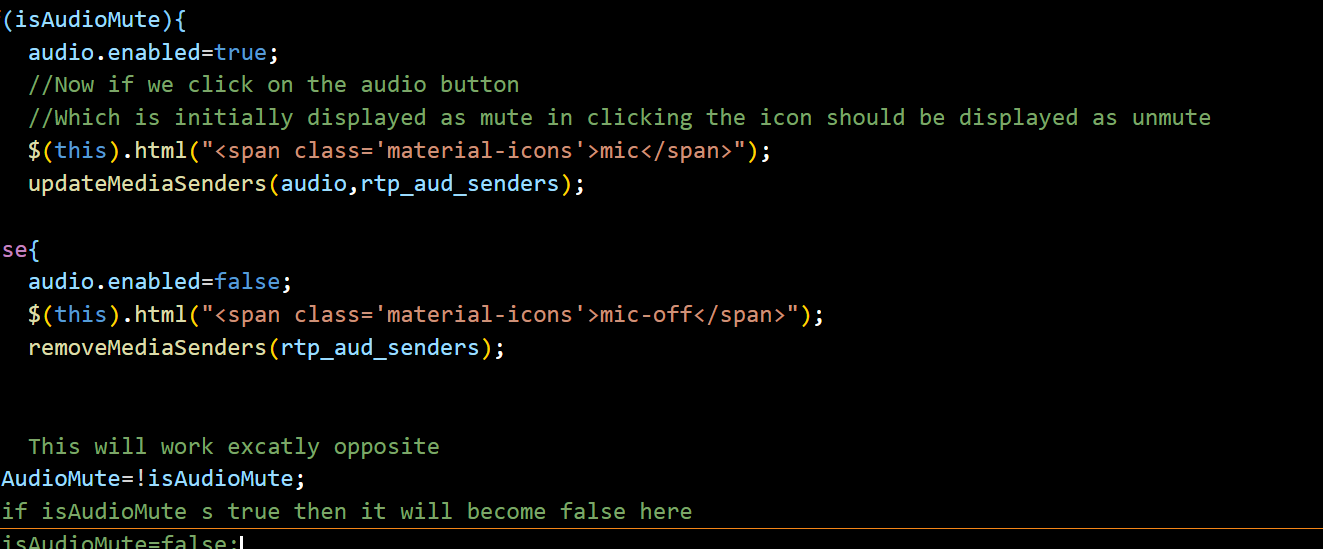


* We will return the connection



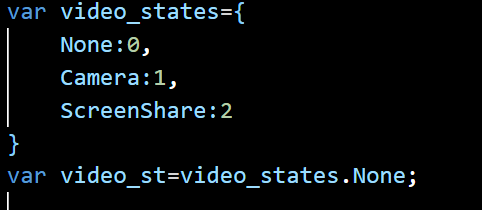
* Now we will try to connect on server side
* Server will get the data and send it to other users
* Now we will write the code to get the sent SDP process data
* 
* Now we need to go to app.js
* 
* In this we will combine processClientFunc to the AppProcess
* So after doing this we have no yet declared the function processClientFunc we will first do it
* 
* Hence to server we are going to return data and from\_connid which our connection id
* Part 5 :Completion of SDP Process function
* In app.js we will create async function of SDPProcess
* 
* 
* Now after returning answer from this it will be checking if it is sent in answer form or not
* We are sending the offer to other users using serverProcess function to SDPProcess now this data is sent to our server
* 
* Part6 informing me about other users
* So in server side we can add
* 
* Now going on client side that is app.js
* 

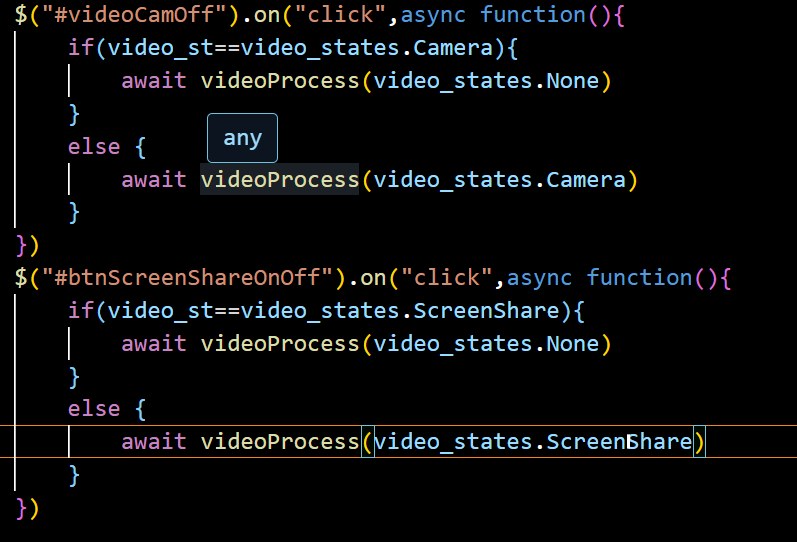
* PART 7 : Creating an option to load audio and vide from the local system
* 
* localVideoPlayer this is same as it was in the index.html
* 
* Initially we will set isAudioMute as true as shown below



Finally, the **isAudioMute** variable is toggled to its opposite value, effectively muting or unmuting the audio for the next click event.

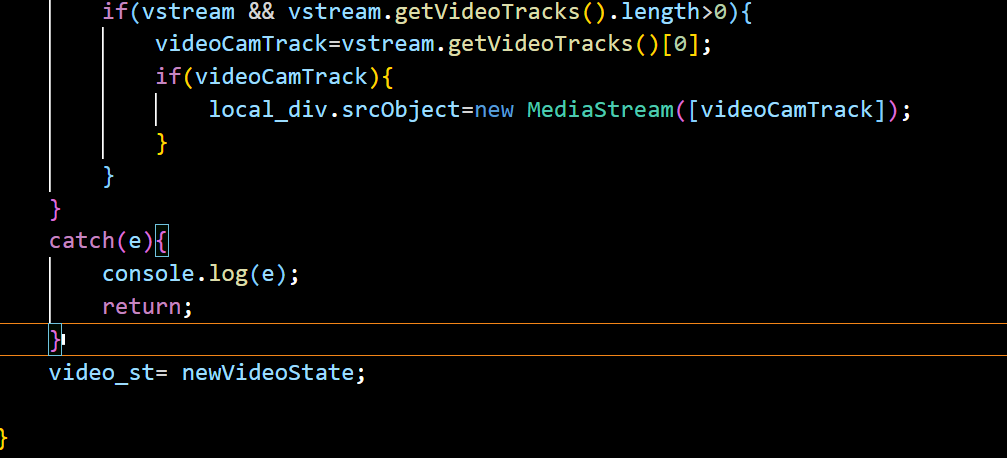
Overall, this code snippet handles the click event of the **micMuteUnmute** element, loads audio if necessary, enables or disables audio based on the current state, updates the UI, and toggles the **isAudioMute** variable.

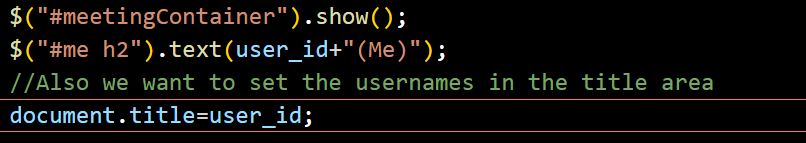
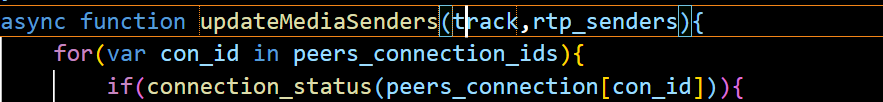
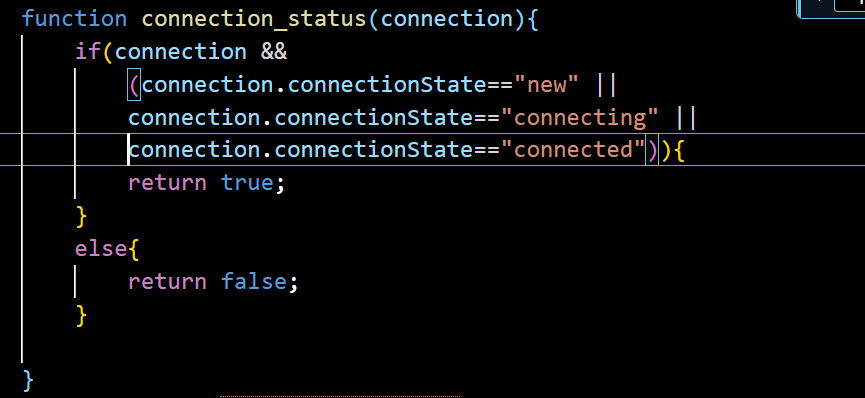
* Now similarly we will do this for video button
* 



PART8: VIDEO PROCESS

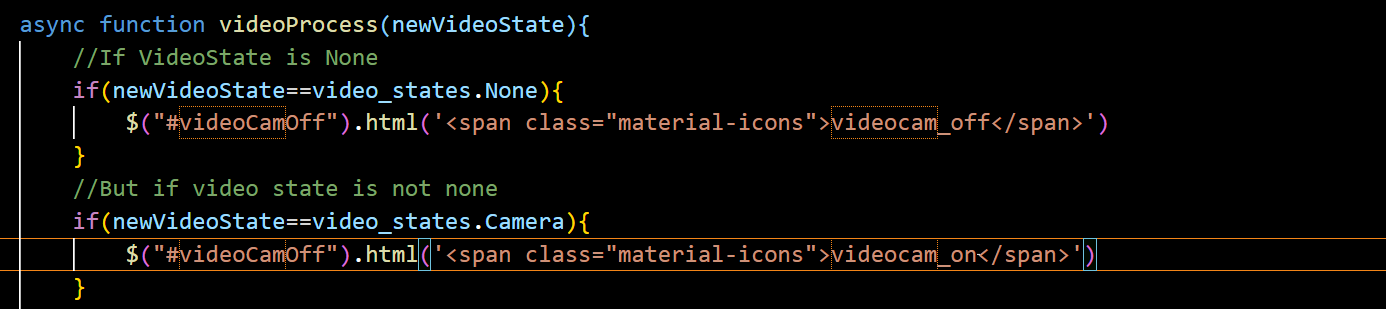
* Now we are going to create the function VideoProcess
* 

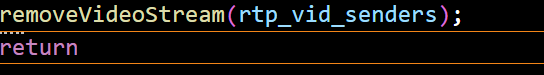


* PART9: Testing our app
* Now here in meeting\_id we had set display none so video is not showing we need to change it to show
* 
* Still this is not working as in index.html we had set that d-none
* 
* Instead of this we will write
* 
* Lec 10 Setting Up the Media Senders
* \*
* Now we will set the function connection\_status
* 

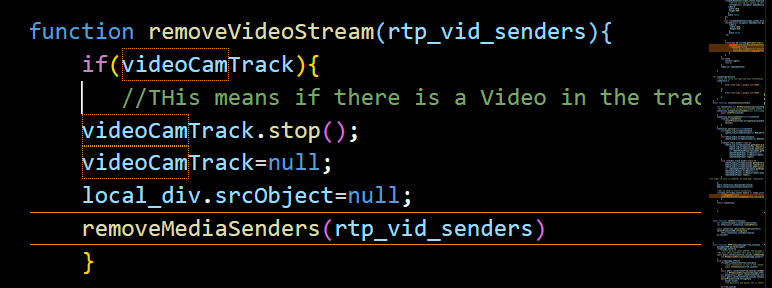
Part 11:

* Now here we are not able to switch off the camera button so now here we will be working on it





* So now we will be creating function removeVideoStream



* Now creating loadAudio function
* 