WEB TECHNOLOGY LABORATORY (22CSL35)

6. Develop a WebRTC (Web Real-Time Communication) platform for avideo calling website using HTML and CSS.

What is WebRTC?

WebRTC, which stands for Web Real-Time Communication, is an open-source technology andset of APIs (Application Programming Interfaces) that enables real-time communicationdirectly between web browsers or other compatible applications. It enables peer-to-peercommunication, such as audio and video conferencing, as well as data sharing, without requiring users to install additional plugins or software.

WebRTC is built into modern web browsers like Google Chrome, Mozilla Firefox, and Microsoft Edge, making it possible for developers to create applications with real-time communication features directly within the browser. It leverages a combination of audio and video codecs, networking protocols, and JavaScript APIs to facilitate secure and efficient communication between devices.

Key features of WebRTC include:

1. **Audio and Video Communication**: WebRTC allows for high-quality audio and videostreaming between browsers, facilitating real-time conversations and video conferencing.
2. **Peer-to-Peer Data Channel**: In addition to audio and video, WebRTC provides a datachannel that allows browsers to exchange arbitrary data directly, making it suitable forfile sharing, gaming, and other interactive applications.
3. **Encryption and Security**: WebRTC incorporates encryption mechanisms to ensure the security and privacy of communications, making it suitable for sensitive conversations.
4. **Cross-Platform Compatibility**: Since WebRTC is supported by major web browsers,developers can create cross-platform applications that work seamlessly on various devices and operating systems.
5. **No Plugins or Downloads**: Unlike traditional communication technologies that often require users to install plugins or software, WebRTC is natively supported by modern browsers, simplifying the user experience.

WebRTC has a wide range of applications, including:

- Video conferencing and online meetings
- Voice and video calls in web applications
- Real-time gaming and collaboration
- Live streaming and broadcasting
- Remote desktop sharing and support
- Internet of Things (IoT) applications

WebRTC has significantly impacted the way real-time communication is implemented on theweb, enabling developers to create rich and interactive experiences without the need for third-party plugins or complex setups.

What is an SDK?

SDK stands for software development kit, also known as a devkit, the SDK is a set of software-building tools for a specific platform, including the building blocks, debuggers and often, a framework or group of code libraries such as a set of routines specific to an operating system (OS).

A typical SDK might include some or all of these resources in its set of tools:

- **Compiler**: Translates from one programming language to the one in which you willwork
- **Code samples**: Give a concrete example of an application or web page
- **Code libraries (framework)**: Provide a shortcut with code sequences that programmers will use repeatedly
- **Testing and analytics tools**: Provide insight into how the application or productperforms in testing and production environments
- **Documentation**: Gives developers instructions they can refer to as they go
- **Debuggers**: Help teams spot errors in their code so they can push out code that worksas expected

Often, at least one API is also included in the SDK because without the API, applications can’trelay information and work together.

Steps for creating WebRTC

Step-1: Setup app on agora.io and get app credentials like Token, App ID and Channel name.

- Go to agora.io and create an account (https://agora.io/). Agora gives us 10000 freeminutes to use each month.
- After login, go to project console dashboard and select the projects tab.Create a new project by filling the basic information needed.
- For authentication select “Secured mode: APP ID and Token”.
- Once your project is created go back to project tab and click on configure. Get your APP ID and Temporary token here by typing your channel name (it can be anything).
- The temporary token generated will last only for 24 hours, generate a new token whenit gets expired or else there will be error on the code.
- In a production environment we will want to generate this token dynamically so users of our app can generate their own channel names to host a call.
- Download the agora SDK (https://docs.agora.io/en/All/download...) from the agora dashboard. For the platform choose “Web SDK” then choose the “Video SDK”. This will download a zip file to your computer, extract the file we need “AgoraRTC_N- 4.18.2.js” and place it directly into our project. This file should be directly pointed to in the index.html file.

Step-2: Creating a folder containing HTML, CSS and JavaScript along with agora SDK.

- Once our credential variables are set in main.js, we use these values to create a client object. The client object is an interface for providing the local client with basic functions for voice and video calls joining a channel, publishing our tracks or subscribing to other tracks.
- Pass in the required properties “mode: rtc” which specifies the optimization algorithmthat will be used and “codec” which is the codec of the web browser will use for encoding.
- Below the client object set some values to represent local video and audio tracks alongwith the values to hold the remote user’s video or audio tracks.
- Local tracks will store user’s video and audio track in a list while all other users that join our stream will be called remote users and this will simply be an object.

- Create a function to toggle local user to join a stream with our camera and audio trackand make sure it is asynch function.
- In the function call the join method from the client object, this method takes in all ourapp credentials and adds our local user to the channel while returning back a UID.
- The local track variable is now a list which holds the audio track in index 0 and videotrack in index 1.

Step-3: Display remote users.

Step-4: Adding controls and styling to our website.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>Web Real-Time Communication WebRTC</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
</head>

<body>
    <button id="join-btn">Join Stream</button>
    <div id="stream-wrapper">
        <div id="video-streams"></div>
        <div id="stream-controls">
            <button id="leave-btn">Leave Stream</button>
            <button id="mic-btn">Mic On</button>
            <button id="camera-btn">Camera on</button>
        </div>
    </div>
```

```
</body>
<script src= “AgoraRTC_N-4.7.3.js” </script>
<script src='main.js'></script>
</html>
```

main.css

```
body{
    background:#0F2027;
    background:-webkit-linear-gradient(to right, #2C5364, #203A43, #0F2027) ;background: linear-
    gradient(to right, #2C5364, #203A43, #0F2027);
}

#join-btn{ position:
    fixed;top:50%;
    left:50%;
    margin-top:-50px;
    margin-left:-100px;font-
    size:18px; padding:20px
    40px;
}

#video-streams{
    display:grid;
    grid-template-columns: repeat(auto-fit, minmax(500px, 1fr));height: 90vh;
    width: 1400px;
    margin:0 auto;
}

.video-container{ max-
    height: 100%;
    border: 2px solid black;
```

```
    background-color: #203A49;
}

.video-player{ height:
    100%;
    width: 100%;
}

button{
    border:none;
    background-color: cadetblue;
    color:#fff;
    padding:10px 20px;font-
    size:16px; margin:2px;
    cursor: pointer;
}

#stream-controls{
    display: none;
    justify-content: center;
    margin-top:0.5em;
}

@media screen and (max-width:1400px){#video-
    streams{
        grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));width: 95%;
    }
}
```

main.js

```
const APP_ID = “YOUR APP ID”
const TOKEN = “YOUR TEMP TOKEN”
const CHANNEL = “YOUR CHANNEL NAME”
const client = AgoraRTC.createClient({mode:'rtc', codec:'vp8'})

let localTracks = [] let
remoteUsers = {}

let joinAndDisplayLocalStream = async () => {client.on('user-
    published',        handleUserJoined)        client.on('user-left',
    handleUserLeft)
    let UID = await client.join(APP_ID, CHANNEL, TOKEN, null) localTracks = await
    AgoraRTC.createMicrophoneAndCameraTracks()

    let player = `<div class="video-container" id="user-container-${UID}">
                    <div class="video-player" id="user-${UID}"></div>
                </div>`
    document.getElementById('video-streams').insertAdjacentHTML('beforeend', player)localTracks[1].play(`user-
    ${UID}`)
    await client.publish([localTracks[0], localTracks[1]])
}

let joinStream = async () => {
    await joinAndDisplayLocalStream() document.getElementById('join-
    btn').style.display = 'none' document.getElementById('stream-controls').style.display =
    'flex'
}

let handleUserJoined = async (user, mediaType) => {
    remoteUsers[user.uid] = user
    await client.subscribe(user, mediaType)
```

```
    if (mediaType === 'video'){
        let player = document.getElementById(`user-container-${user.uid}`)if (player != null){
            player.remove()
        }

        player = `<div class="video-container" id="user-container-${user.uid}">
                        <div class="video-player" id="user-${user.uid}"></div>
                </div>`
        document.getElementById('video-streams').insertAdjacentHTML('beforeend', player)user.videoTrack.play(`user-
        ${user.uid}`)
    }

    if (mediaType === 'audio'){
        user.audioTrack.play()
    }
}

let handleUserLeft = async (user) => {delete
    remoteUsers[user.uid]
    document.getElementById(`user-container-${user.uid}`).remove()
}

let leaveAndRemoveLocalStream = async () => {for(let i = 0;
    localTracks.length > i; i++){
        localTracks[i].stop()
        localTracks[i].close()
    }

    await client.leave()
    document.getElementById('join-btn').style.display = 'block'
    document.getElementById('stream-controls').style.display = 'none'
    document.getElementById('video-streams').innerHTML = '  '
}
```

```
let toggleMic = async (e) => {if
    (localTracks[0].muted){
        await localTracks[0].setMuted(false) e.target.innerText =
        'Mic on' e.target.style.backgroundColor = 'cadetblue'
    }else{
        await localTracks[0].setMuted(true) e.target.innerText =
        'Mic off' e.target.style.backgroundColor = '#EE4B2B'
    }
}

let toggleCamera = async (e) => {
    if(localTracks[1].muted){
        await localTracks[1].setMuted(false) e.target.innerText =
        'Camera on' e.target.style.backgroundColor = 'cadetblue'
    }else{
        await localTracks[1].setMuted(true) e.target.innerText =
        'Camera off' e.target.style.backgroundColor = '#EE4B2B'
    }
}

document.getElementById('join-btn').addEventListener('click', joinStream)
document.getElementById('leave-btn').addEventListener('click', leaveAndRemoveLocalStream)
document.getElementById('mic-btn').addEventListener('click', toggleMic)
document.getElementById('camera-btn').addEventListener('click', toggleCamera)
```