Mini Project- Report

Aug-2022-2023

Course Faculty : Prof. Sahana MP Course Name & code : Mini Project

Semester : 6th Date : 04-07-2022

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TITLE OF THE PROJECT | ChitChat - WebRTC Video Calling App | | | |
|  |  | | | |
| STUDENT NAME | Xitiz Verma | Yashaswini Shree | Mohammed Shagil Khan | Soumya Kumari |
| USN | 1DS19CS193 | 1DS19CS196 | 1DS20CS415 | 1DS19CS161 |
| INDIVIDUAL  CONTRIBUTION | Foundation Implementation and Structuring of System Code. | Programming, UI Design and Framework Layout  Compartmentalizing ,Bootstrapping. | Researching WebRTC SocketIO, Connectivity of ICE Candidates in SDP Debugging. | FireBase Signalling Server Configuration and  RTC Object & Database Setup. |
| GUIDE | PROF. SAHANA MP | | | |
|  |  | | | |
| PROJECT ABSTRACT : | **In this age of social distancing, Let's Chitchat.**  WebRTC (Web Real-Time Communications) is an open source project which enables real-time communication of audio, video and data in Web and native apps. And we’ll use Socket.IOand Node Js Development Services for setting up signaling server.  WebRTC is a new standard for enabling Real Time Communication (RTC) within a web browser. A web browser that has support for WebRTC includes the necessary technology to build a two-way video chat client directly in the browser without requiring the user to download any software.  The WebRTC project was initiated by Google and standardization is being performed both at W3C and the IETF. WebRTC is being rapidly adopted by numerous technology companies. | | | |
|  |  | | | |
| Introduction | With the advent of WebRTC and the increasing capacity of browsers to handle peer-to-peer communications in real time, it’s easier than ever to build real-time applications. In this article, we’ll take a look at SimpleWeb RTC and how we can use the platform in implementing WebRTC technology. We’ll also look at other alternatives that could help us achieve the same goal.  webrtc stun only signaling  To set up WebRTC signaling, the ICE framework requires you to provide two types of servers, detailed below.  **1. STUN Server**  The **STUN (Session Traversal Utilities for NAT)** server does exactly what I’ve just described above. It simply provides a meeting space for computers to exchange contact information. Once the information is exchanged, a connection is established between the peer computers and then the STUN server is left out of the rest of the conversation.  Here’s an example script that runs on the client, which allows the browser to initiate connection through a STUN server. The script allows for multiple STUN server URLs to be provided in case one fails.  Connections established via STUN servers are the most ideal and cost-effective type of WebRTC communication. There’s hardly any running cost incurred by the users. Unfortunately, the connection may fail to establish for some users due to the type of NAT device each peer is using. In such a situation, the ICE protocol requires you to provide a fallback, which is a different type of signaling server known as a **TURN** server.  **2. TURN Server**  A **TURN (Traversal Using Relay NAT)** server is an extension of the STUN server. Where it differs from its predecessor is that it handles the entire communication session.  Basically, after establishing a connection between the peers, it receives streams from one peer and relays it to the other, and vice versa. This type of communication is more expensive and the host has to pay for the processing and bandwidth load required to operate a TURN server.  Below is a graphical depiction of the entire signaling process involving first the STUN server and then the TURN server as fallback. webrtc stun turn signalling  Above is a complete Architectural Diagram showing the entire WebRTC process. | | | |
|  |  | | | |
| Design | We successfully implemented the video calling App wherein more than two people can currently video call together using the web app. We also implemented:  1. Switch audio on/off  2. Switch video on/off  3. Create your own meeting url  4. Join an existing meeting with more than two people  5. Adding own name to profile with which you join the meeting  6. Update whenever somebody joins or leaves the meeting  7. Copy URL in-meeting  8. Chatbox  9. Screenshare  10. Collaborative whiteboard  11. Leave meeting | | | |
| PLATFORM USED  (H/W & S/w tools to be used | React Js, Firebase Signalling Server,  WebRTC Peer Connection API, Socket.IO, Node, ExpressJs,  CSS BootStrap | | | |
|  |  | | | |
| Project Source Code Link (Github/ Google DRive) | *https://github.com/XitizVerma/Chitchat---Video-Calling-WebRTC-App* | | | |
|  |  | | | |
| Conclusion /FUTURE ENHANCEMENT | To extend Support Components for even more versatile funtionality.  We will deploy Chitchat over Heroku so that our fellow Community can use it. | | | |
|  |  | | | |
|  |  | | | |

**UI SCREENSHOTS**

