**Audio/Video Communication Application using WebRTC**



**Department of Electrical Engineering**

**Abstract**

WebRTC is a project that allows users to establish any real-time connection of audio, video or data form their web browsers. This project gives control to the end user to create and modify a real-time communication channel from their web browsers has a very wide scope of application. Since this is an Open Source Project that is trying to allow end users of RTCP to create and modify their channel. It comes with a set of predefined API’s. Using these API’s end-user can modify and create a real-time peer to peer connection from one web-browser to another.

**Introduction**

There are plenty of applications in the market that are used nowadays for video calling and conferencing. But the main disadvantage behind them is that we need to download the application or the plugin before we can actually establish a video call or a conference. These applications are build using complex codes, various protocols and have got plenty of things that are running on the backend of the application. This all is simplified with WebRTC. The need to download the applications is no longer necessary. All you need is a web browser which is provided by default in all the mobile devices, computers and other devices.

WebRTC is basically web based real-time communication. It is a collection of communication protocols and APIs (application programming interface). Together they enable real-time communication between two peers without the need to install or download any plug-ins. Along with video and audio , text messages and files can be transferred without any external or internal plug-ins. WebRTC is being standardized by the World Wide Web Consortium(W3C) and the Internet Engineering Taskforce(IEFT) [5]. WebRTC is low cost and provides high quality audio and video communication along with file sharing feature. WebRTC supports lots of platforms like chrome, FireFox, Opera, etc. There are several tasks associated with WebRTC to start a real -time communication. Initially, the browsers of the two people must agree to start communication, know how to locate one another, bypass firewalls and defenders in order to establish communication [6].

The main goal of our project is to develop an application that can carry out real time audio and video communication.

**APIs used**

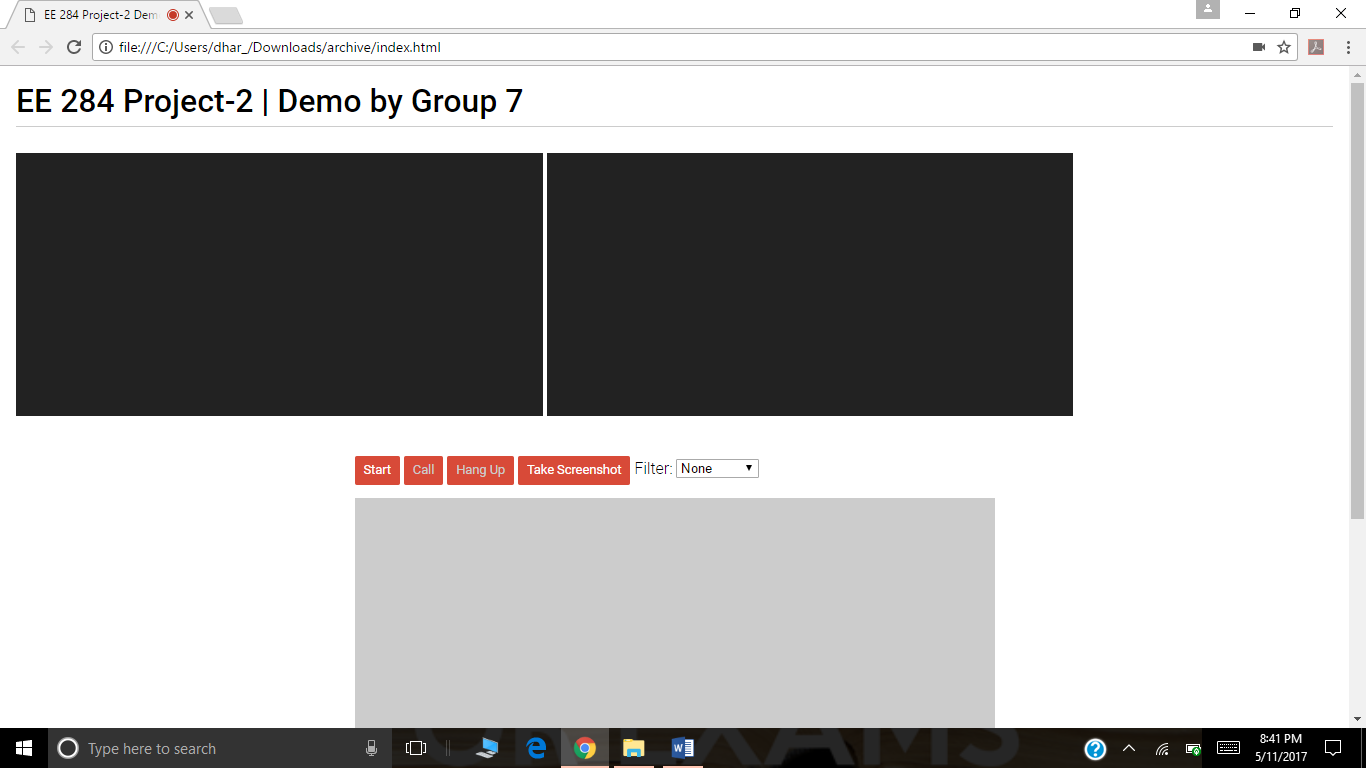
1. **getUserMedia()** : The API that allows the end user to interface its hardware for the communication, like mike, display, webcam and keyboard input/output. It gives control to the user to allow or deny the permission for any I/O on the device, for communication. The it allows fine grain control of these devices. Like we can manage the volume of the audio input or output, the resolution of the video that is to be transmitted for the commination, the display size for a video communication. The services provided by this API at the end improves the end user experience, since each user can make modifications according to their preference or need.
2. **RTCPeerConnection()** : It is the API used to initiate and establish a real-time connection with a peer. It is responsible for the signaling, session establishment and session termination. It allows the user to manage the bandwidth that will be used for the communication. It also enable a user to relay multiple data streams as one, to efficiently use the channel.

**Project Implementation**In order to establish audio and video real-time communication we did the following steps:

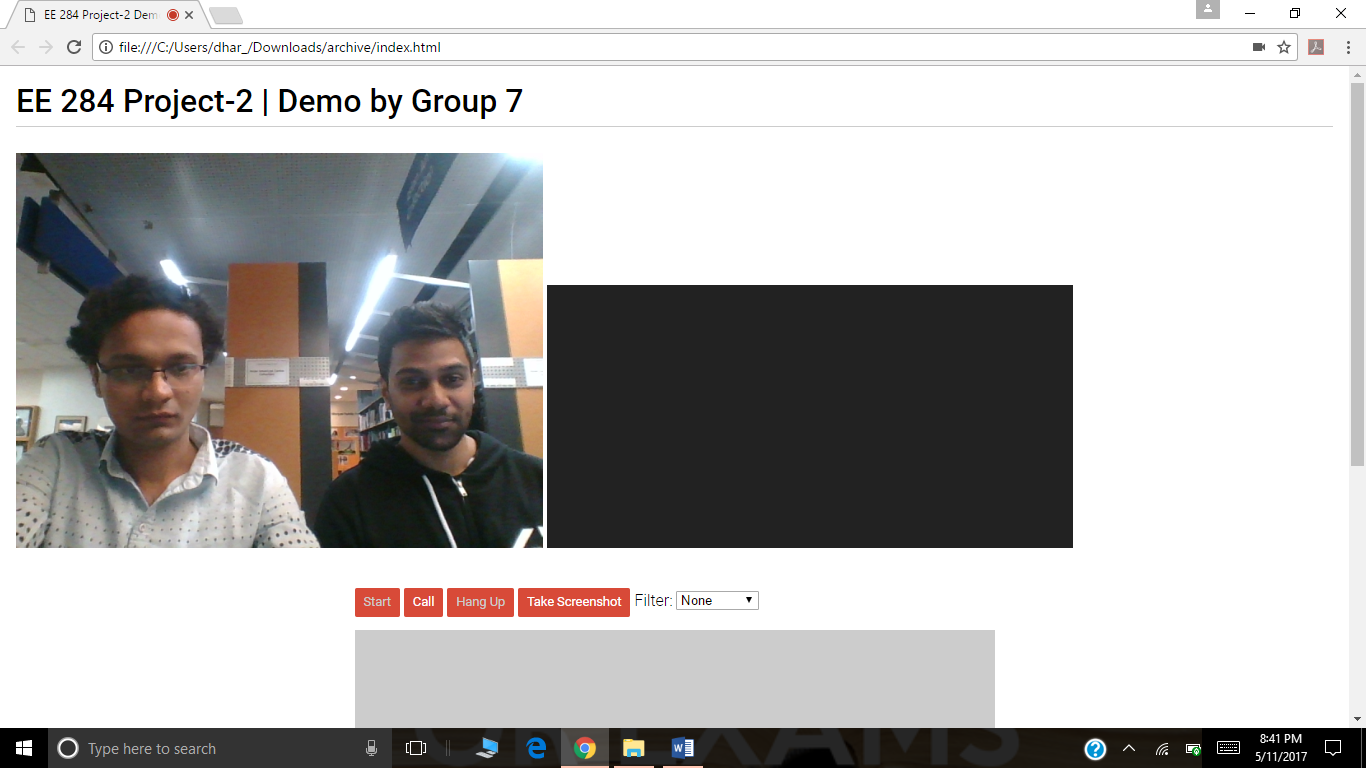
1. Actually, STUN and TURN servers are used in the real scenario to connect via best possible direct route [2]. We have implemented the communication on the same browser.
2. Wrote a basic html script to show the video windows with Start, Call, HangUp and Take Screenshot buttons.
3. Then, we developed a Java Script to implement the audio/video real-time communication using the getUserMedia and RTCPeerConnection.
4. We have imported the java script file named adapter.js from Github. The adapter.js is a shim to insulate applications from spec changes and prefix differences [7].

**Results**

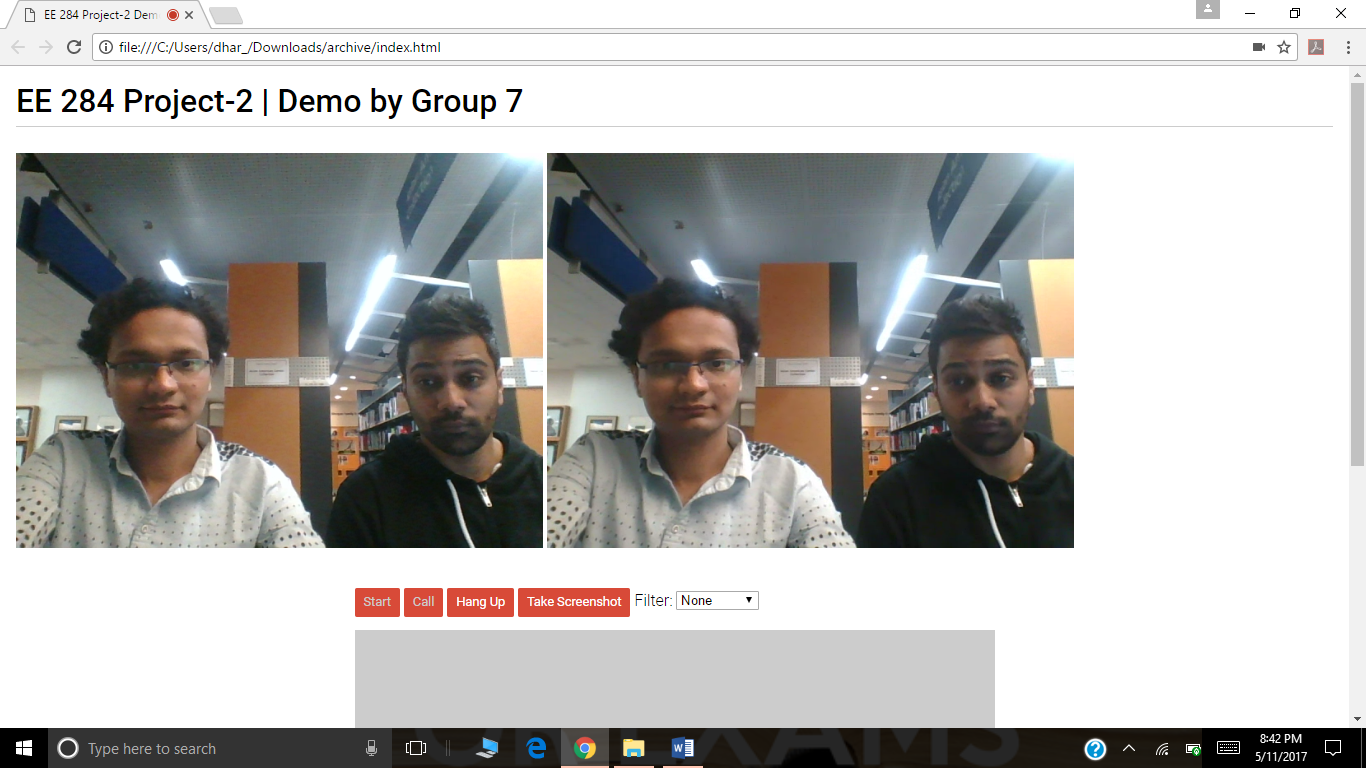
1. The screenshot below shows a basic webpage which we created for testing purpose with four buttons: start, Call, HangUp, Take Screenshot and filters dropdown list.



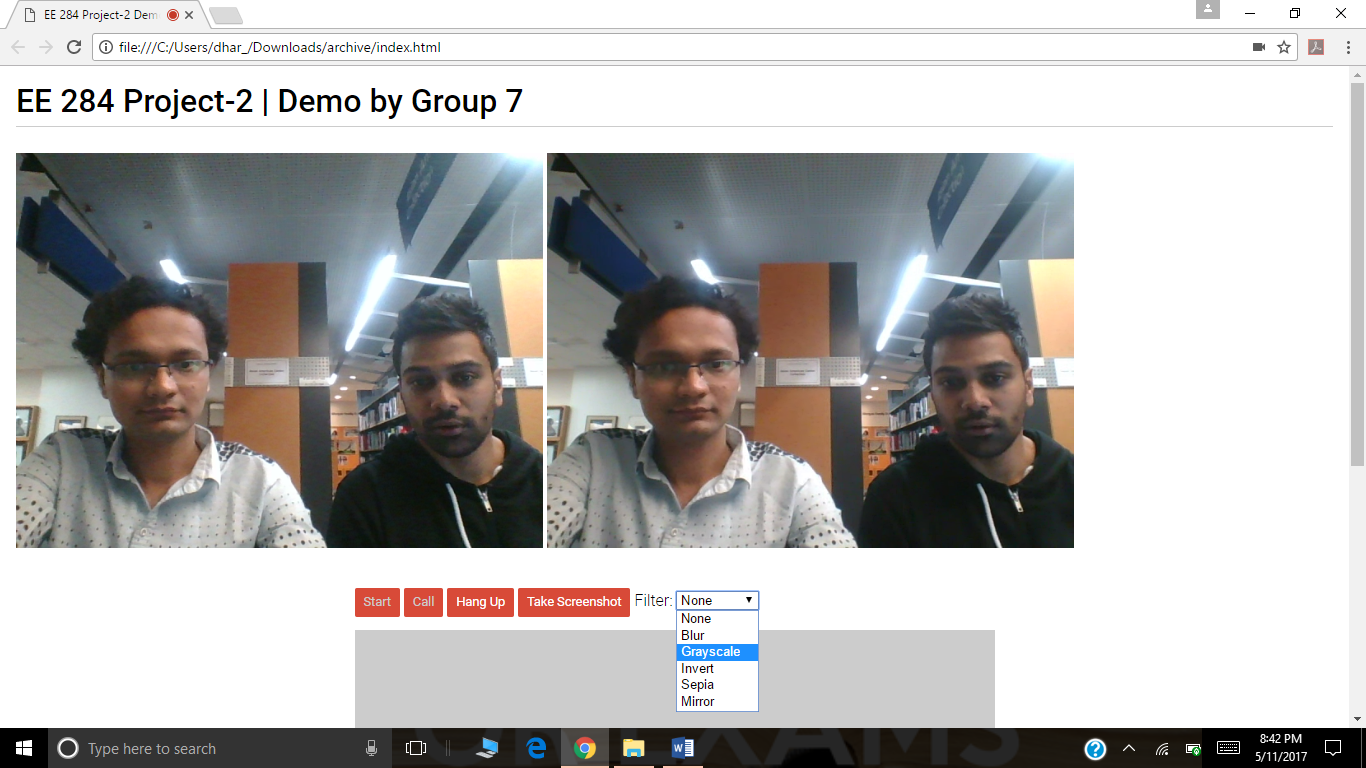
1. After clicking the start button the video for the user is started, we can see that in the screenshot below.

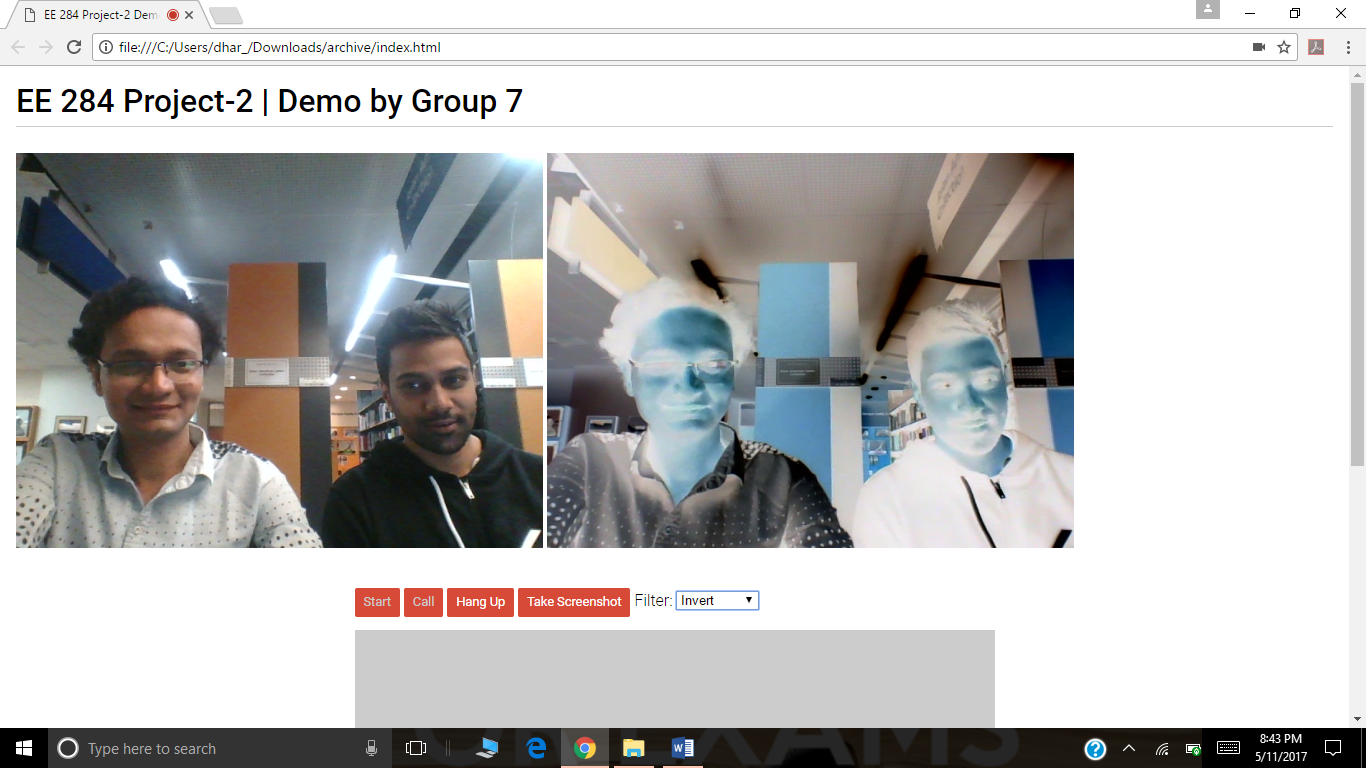


1. And eventually when we click the call button, audio/video call is established between two users.  
   Note: For testing purpose, we have written the java script for communication between clients on the same browser.

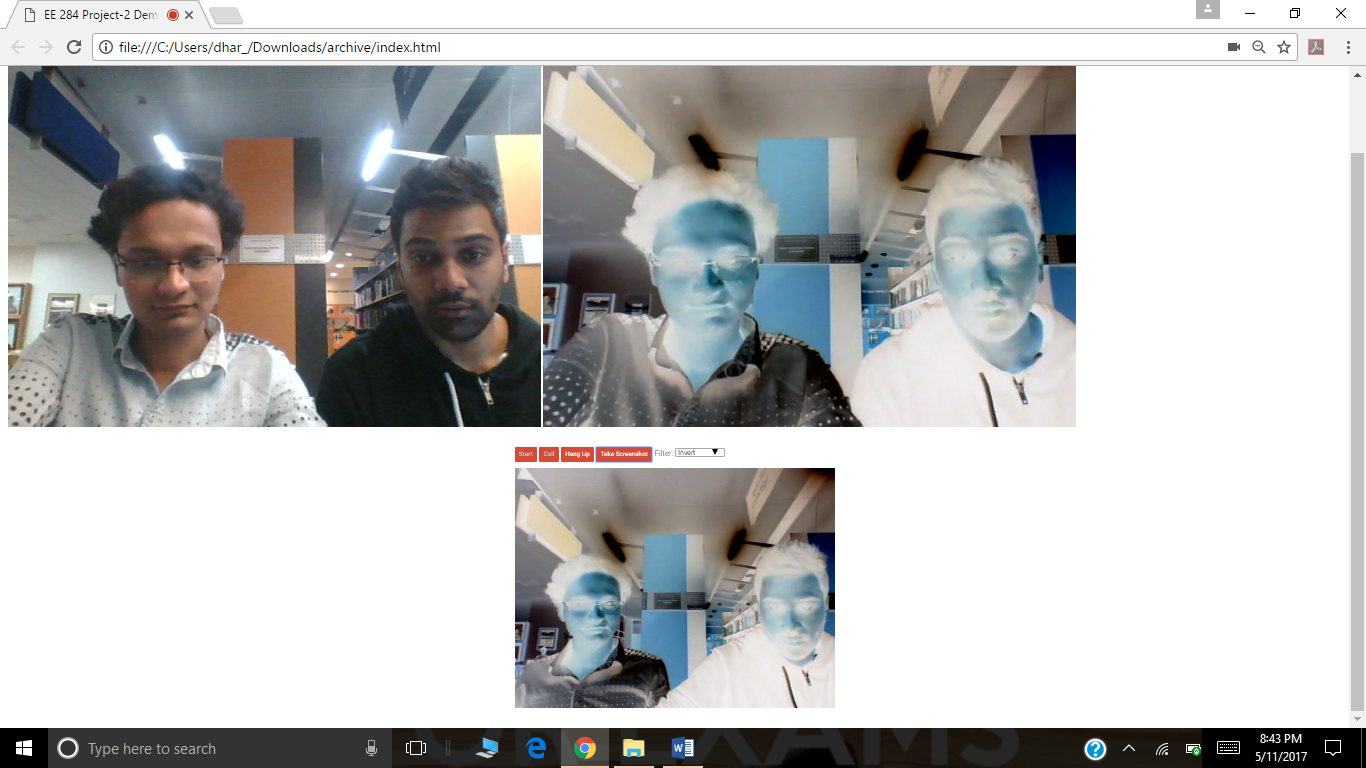


1. We have also added functionality of video filters. There are five filters, viz, Blur, Grayscale, Invert, Sepia and Mirror.





1. When Invert filter is selected, if one clicks the “Take Screenshot” button, an image of the remote user video is taken and displayed in the canvas.



**Future Scope**

The scope of the project we have implemented can be further extended by establishing audio/video call between two browsers(users) on different computers. For implementing that we need to further figure out a way to locate the browser with which you want to communicate with and establish a network socket connection with that user’s web browser. Adding file sharing functionality would be a good upgrade along with text messaging feature. In future, we can implement a STUN/TURN server to get a peer-to-peer connection. STUN is Session Traversal of User Datagram Protocol through Network Address Translators server. It allows different IP addresses that is devices to setup an audio/video or audio and video connection so that two different clients from anywhere in the world can connect each other and start a VoIP call. In added features, we can add some filters by using image processing concepts or can make it more interactive by changing its User Interface

**Conclusion**

By implementing this project, we learned how audio/video communication can be established between two browsers using the getusermedia and RTCPeerConnection APIs. Also, we got the deep understanding of how WebRTC actually works.

**References**

1. *Io13webrtc.appspot.com*. N.p., 2016. Web. 18 Nov. 2016.
2. "Getting Started With Webrtc - HTML5 Rocks". *HTML5 Rocks - A resource for open web HTML5 developers*. N.p., 2016. Web. 25 Nov. 2016.
3. "Real Time Communication With Webrtc". *Codelabs.developers.google.com*. N.p., 2016. Web. 2 Dec. 2016.
4. "Building A Video/Audio Chat Web App With Webrtc | Web-Engineering.Info". *Web-engineering.info*. N.p., 2016. Web. 7 Dec. 2016.
5. "Webrtc". *En.wikipedia.org*. N.p., 2016. Web. 13 Dec. 2016.
6. Project 2.pdf provided on canvas.
7. "Webrtc/Adapter". *GitHub*. N.p., 2016. Web. 13 Dec. 2016.
8. “Webrtc Samples”. https://webrtc.github.io/samples/