Team Viewer Project  
  
Presented by:  
Sameh Mohamed  
Ahmed Nabil  
Hazem Sherif  
Aly Barakat

Application Description  
  
The application is remote desktop, similar to team viewer. The protocol used is WebRTC.  
  
To begin with, signaling takes place, which is important to establish a connection and for both devices to locate the other.  
The process involves both devices connecting to a third, mutually agreed-upon server. Through this third server, the two devices can locate one another, and exchange negotiation messages.  
Where one client requests connection “offer” and the other replies with their data “answer”  
  
Once we are done with signaling, we can establish an RTC Peer to Peer connection, which connects the ICECandidates to each other, this is done via WebSockets.  
This is a very useful webRTC tool to allow the exchange of data and media , only one parameter is given to the function, which includes the URL of information about the ICECandidates you want to connect to each other.  
  
To stream and transfer media, getUserMedia() is used.  
It is also a webRTC tool for streaming video and audio, based on the given constraints. It could stream screen sharing or video chat or just an ordinary video, returning the data type of MediaStream which is frames of audios and video.  
In this case, it is used to implement screen sharing.  
  
WebRTC is implemented by JavaScript, while the GUI is implemented by HTTP, CSS.  
The frameworks used is Electron, which is also supported by NPM JavaScript.

Code Description  
  
Variable of **UserData** consists of a captcha which is randomly generated everytime, and a username that is enternet.  
  
**Conn.OnOpen**: Confirmation that the signaling server is working.

**OnMessage**: Based on the message type, whether it is an answer or an offer.  
  
**LoadStart**: On login of the user, he is assigned a username, and it is logged on the console.  
  
**HandleLogin**: First, the **userMedia** function is called which captures the screen and assigns it to the HTML video element.  
Then a peer to peer connection is established, and data is streamed over that connection. This is done by the built in **PeertoPeer and icecandidate** event handlers.  
  
**CreateOffer:** When the create button is clicked, a message of type offer is created, this is useful for the **OnMessage** function as the message type is a used parameter.  
  
**HandleOffer:** To handle the offer, you need to create an answer which is the response to establish a connection, therefore the **CreateAnswer** is function is then called.   
  
**HandleLeave**: On logout, everything is reset to null.