**SUMMARY OF VIDEO STREAMING APP PROJECT**

The app built is a streaming socket network program consisting of a client program and a server program. The server program sends a video to the client program while the video runs on both ends. For this to work it is required that the server and client sides are connected to the same Wi-Fi. It is also required that the server and client programs have the same port number so that the client can access the incoming video from the server. This app uses UDP instead of TCP. UDP is the connectionless protocol while TCP is connection based. The nature of UDP which makes it more preferrable than TCP here is that in order to guarantee reliable transfer will wait until a missing packet is received before the next transmission. This would cause longer than normal pauses in the video stream. Therefore, UDP is preferred as it continues sending packets regardless of where they have been received on the client end or not. Our application is an example of what is called a time-sensitive application which is one where delays (normally over a second) in transmissions aren’t tolerated. This makes UDP a better fit in such applications. This program streams the video by sending successive image frames from server to the client such that the video is played in real-time simultaneously on both ends. Below is the summary of the steps the client and server sides of the program streams a video successfully:

**Server**

1. Initialize buffer size of each image frame.
2. Create server connection
3. Create a loop that continually sends image frames of video to client
4. Within the loop in 3:
5. Resize image frame to one that fits within given buffer size
6. Convert image data into streaming data which is in binary
7. Convert binary data into text
8. Send frame to client
9. Display transmitted frame with frame rate displayed at top of image.
10. Add a way to close connection when user of server wishes to.
11. Calculate frame rate.
12. Loop ends

**Client**

1. Initialize buffer size of each image frame.
2. Create client connection.
3. Create a loop that continually receives image frames of video to client.
4. Within loop
5. Receive datagram from server.
6. Decode image frame data from received datagram.
7. Convert decoded data into a 1d array.
8. Convert 1d array into an image and add text to the image which indicates frame rate.
9. Add a way to close connection when user of client wishes to.
10. Calculate frame rate.
11. Loop ends