**Description of Program:**

The following program should run three program: Controller, Server, Renderer. That being said, all three are connected by mininet. Controller is used to controller what happens to the renderer. If Controller ask for the renderer to play said “file”, then server would send the file contents to the renderer, and then renderer will play and display it through its console. The Server is there for holding files and taking request from the Controller, if Controller ask for file or ask for what files are available the Server sends it to the Controller or redirect the contents to the Renderer to play. As for the Renderer, the renderer plays the files by Controller Request, and the functions pause, play, and start from beginning are used by request of the Controller.

**Challenges:**

The hardest part of the project wasn’t the project itself. It was working with others, while me and my partner Jonny were hashing out what needed to be done, the other two group members did nothing. We set up meetings every Monday, no one really committed to that, so Jonny and I started to come every week on Wednesday to work on the project. I told that to our group members, but not having text proof they denied the clause. Even though we talked about it after class, saying that “me and Jonny work on the project every Wednesday if you guys wanted to join us.” If that wasn’t enough neither of those said group members initiated any group meetings nor talked about the project. That itself lead us to disbanding the group and splitting into two groups. Also because when we were almost done with the project, they both then wanted to take part on the last week of the project. Trying to take credit for work they did not produce. If it makes my case any different, a week prior to them wanted to take credit I asked them to at least look at the github, to understand what we had. In return, they did not do at all.

**What I learned:**

Group projects are hard. Socket programming is easy once you understand what is needed to be done. TCP is the easiest way to do this project, because you’re depending on a reliable way to send information. UDP would be better only in a case where you are streaming a video(in this case we did not do). If I were to do this in java it’d take a longer time also. If you’re trying to sending information to two clients you can create two sockets and two ports to differentiate between the two clients, but you have to make sure you listen for those two socket connections.

The biggest thing I learned was: How to use python. Python I thought was tough because I’ve never wrote anything but scripts for this language, but it turns out it wasn’t as hard as it could have been if I used a different language.

**a discussion about algorithms and techniques used in the program**:

So the way that the three interact is Client ask server for file, Server says yes or no, Server then sends lines to renderer to display, and Renderer displays the lines from the file. The   
algorithm in this case, server redirects the functions to renderer or client.

**Contributions of each member:**

50% - Jonny Le

50% - Jethjera Silasant

Other two 0%