# How to use

## Configurations

In helperFunctions.py, you can set the prime node by changing the variable PRIME. To set the ip address of your machine (for running other nodes), change the IP variable

For the clientside program, user.py, change your IP address by changing the IP variable. This is used for p2p connections. Additionally, you can change the P2P port by changing the variable P2PPORT

To change the maximum number of nodes a control node (prime node and other control nodes in the control plane), change the variable maxLoad in controlNODE.py

# Server side

To start the server side, run primeNODE.py on the machine that you want to use as the primary control node. This node is responsible for keeping track of all nodes in a list and redirecting requests to them. It also checks the checks how many users are connected to authentication nodes (dbNODE.py) when a user logs in. If the maximum capacity is reached on an authentication node, it will keep iterating through its list of known authentication nodes. If none are found, it will attempt to spawn a new authentication node.

Once a user logs in, it will be redirected to a service node (fileNODE.py) in order to obtain a list of files to be streamed. Whilst logged in, the authentication node will maintain a heartbeat to users to keep them authenticated. After a service node or authentication node completes an operation, it will check to if users are still connected. If not, they will shutdown automatically, freeing up ports on the machine. Furthermore, only authenticated users can stream music through service node as it checks all authentication nodes to make sure the heartbeat is still active.

When the prime node reaches its node capacity, it will attempt to find another control node to request to spawn the required node. To add a control node to control plane, on a different machine run, planeNODE.py. When a control node disconnects from the prime node, all nodes that share the disconnected nodes ip address are removed from the list nodes. To avoid manually adding a new WAV file to each machine, the prime node will regularly request a list of files from each control node, sending and downloading files if needed.

# Client Side

Chart

Description automatically generated with low confidence

On the client side, login using any of these usernames:

* Someone
* AnotherUser
* FinalUser

The password for all users is **password123**. Once logged in, you can select on song on the left to play. Whilst playing, you can stop the song by pressing the pause button. (the play and stop don’t do anything as it was originally intended for the system to be able to pause and unpause the stream). If music is being streamed from the service node, a server socket will open on the client, allowing other clients to connect. This enables them to join the stream at the same point that the p2p host is at, bypassing the need for the service node.

# How the system was tested

The server side was tested using multiple computers as planeNODE.py was not intended to be ran on the same device as the prime node. The list of nodes was tested by manually opening different node types and shutting the down to see if the list updated correctly. Although initially, tested on the same machine as the prime node, disconnecting planeNODE.py results in the prime node deleting nodes on its own machine from the list due to mistaking them to be on different machines. The authentication node was tested by logging in with all 3 users. This would result in the authentication node reaching its maximum capacity of 2 and redirecting the third user to the 2nd authentication node as intended.

The service node was tested by having multiple logged in users request to play a song. This was done to ensure one client’s request did not conflict with another. The result was as intended, both users playing different music simultaneously.

To test the file transfer between control nodes, the control node connected to the prime and after a few seconds, a new file was moved into the music file on one the control node, resulting in the prime node detecting the change and requesting to download the file. The reverse was also tested, with the prime node detecting a missing file and sending the required files over.

Lastly, P2P was tested by one user play a song and then another user login and enter the chosen IP and port of the first user. This result allowed p2p clients to join, leave and re-join the stream at the correct point.