100525814 Distributed Systems Assignment 1

Contents

[Testing 3](#_Toc125703223)

[Outcome of System 26](#_Toc125703224)

# Testing

I started by creating a simple server and client that can handle connections. I tested to see if my server and client could connect by trying a hard coded connection.

Text

Description automatically generated

The server and client are not working as intended, I have incorrectly set up the hard coded connection.

Text

Description automatically generated

Text

Description automatically generated

I then set up a unit test file where I will be doing my future feature testing. I made it so the client and server run on the same thread. I then used the same host and port to try to connect them.

Text

Description automatically generated

By doing this I got the first successful connection. As there is no input handling yet the expected response is “Unknown command”.

Text

Description automatically generated

I then refined the test file by setting up a correct unit test for them. I also imported multiprocessing into the file as then I can run multiple processes at once for the individual server files and the client.

Text

Description automatically generated

It passed the first test that I made to test if the client and server could connect. The pass for the test is the response being “Unknown command”.

Text

Description automatically generated

I then started setting up the node which is the combination of the server and client files.

Text

Description automatically generated

I split the files up into the core, server, and client files. This will be useful for testing purposes and separating the server, core, and the client files.

Text

Description automatically generated

I created the Prime Node using the Core Node class as a base as this will be the primary node. I also create base files for the Authentication Node, Control Node and FDS Node. I then created a User Client using client as a base. I will be using the User Client as a way to test each of the server nodes.

The Prime Node will be tested by the nodes connecting to it and sending it a specific command, for the function nodes a register command will be needed and for the user client and FDS node will need commands to get the authentication node and the FDS node from the prime nodes registered nodes list.

Text

Description automatically generated

The prime node was tested to see if a client could connect was through the same process the original server and client connection was tested however the host and port is set to be a known address for all the nodes to connect to.

Text

Description automatically generated

The FDS node has been created so that it will check the token which the user gives to the node before allowing it to fully interact with the node. The FDS Node functions are split into functions so that the overall testing of the node is easier to debug.

Text

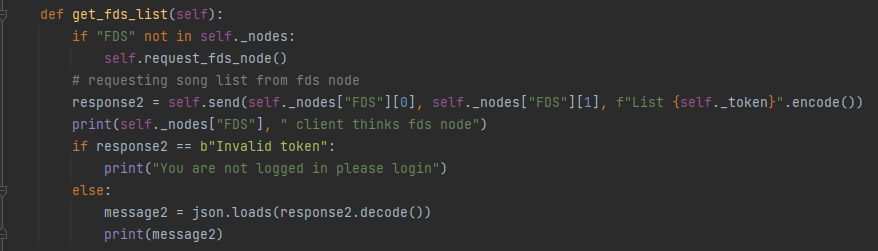
Description automatically generated

The user client is setup so that the testing of it and the testing of the nodes are simplified due to the specific features being separate from one another. The user will be able to enter the general command of what they want to do and there is a function that will handle the command specifically. The connection between the client and the prime node alongside the getting of the FDS and Authentication nodes will be handled in the background when the specific function is called.

Text

Description automatically generated

The FDS will need to check if the client has got the FDS node address stored so that the list function can be tested. The FDS will be returning a json as the song list only if the token from the authentication process is valid, it will be tested if the list is given back to the client though an if statement to handle whether the FDS returned the list or there was an error when trying to get it.

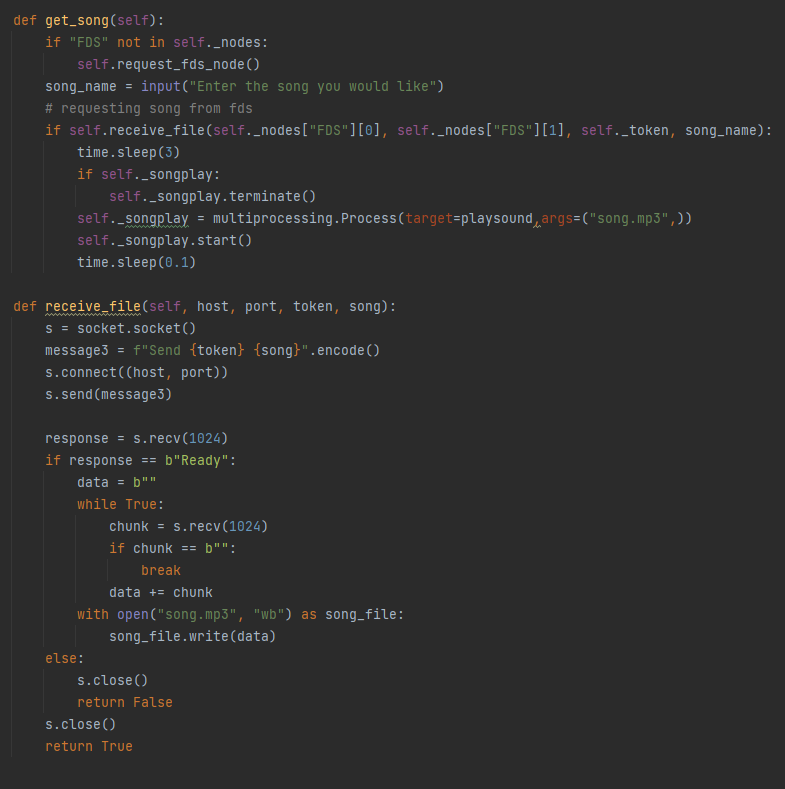


The getting of the FDS list was tested successfully through connecting a client directly to the FDS node and sending it the list command so it returns the song list.

Text

Description automatically generated

The get\_song and receive\_file functions in the client will be used to test the getting and receiving of the song file from the FDS node.



The send function in the FDS node will be responsible for handling the sending of the song file over, it will send the song over for the client to receive. It will send “ready” to the client so that it can get ready to receive the song data.

Text

Description automatically generated

This is the test that was used to all the commands the user can send to the FDS and the getting of the FDS address for the client.

Text

Description automatically generated

The check\_token function in the FDS node will be used to test the token that is given by the client. This function will get the Auth connection from the Prime node and then send the client token to the Auth node to check if its valid.

Text

Description automatically generated

The client function auth node functions, the request\_auth\_node will be used to test the getting of the auth node from the nodes stored in prime node. The register\_with\_auth and login\_with\_auth functions will be used to test the auth node registering a client, logging a client in, and giving them a token.

Text

Description automatically generated

The authentication node is similarly structured to the FDS node, in which there are several functions to handle the main processes of the node. This node will be used to test is a token the client is trying to use is a valid token stored within the authentication node.

Text

Description automatically generated

The login function will be used to test the username and password given by the client and if it is stored within the authentication node. After the client logins in the token function is used to give the client a token to test FDS functions and the Check\_token function in the authentication function.

Text

Description automatically generated

This is to test authentication node functions with the rest of the system. This test successfully passed the testing of using the tokens with the other nodes and the client.

Text

Description automatically generated

I made the control node able to spawn the nodes the prime node needs. This was tested through making prime node ask if a control node is free and if it is free then send the name of the node it needs and in the control node it will spawn whatever node it is sent.

Text

Description automatically generated

Text

Description automatically generated

The successful test for the prime node. It shows that the control nodes have connected to the prime node and the prime node has sent a command for them to spawn the nodes that the prime node requires.

Text

Description automatically generated

The successful test for control node 1. This shows the control node connecting to the prime node, then the prime node asking the control node if the node is free and then turning it into an auth node because it is a free node.

A picture containing text, plaque

Description automatically generated

The successful test for control node 2. This shows the control node connecting to the prime node, then the prime node asking the control node if the node is free and then turning it into an FDS node because it is a free node.

Text

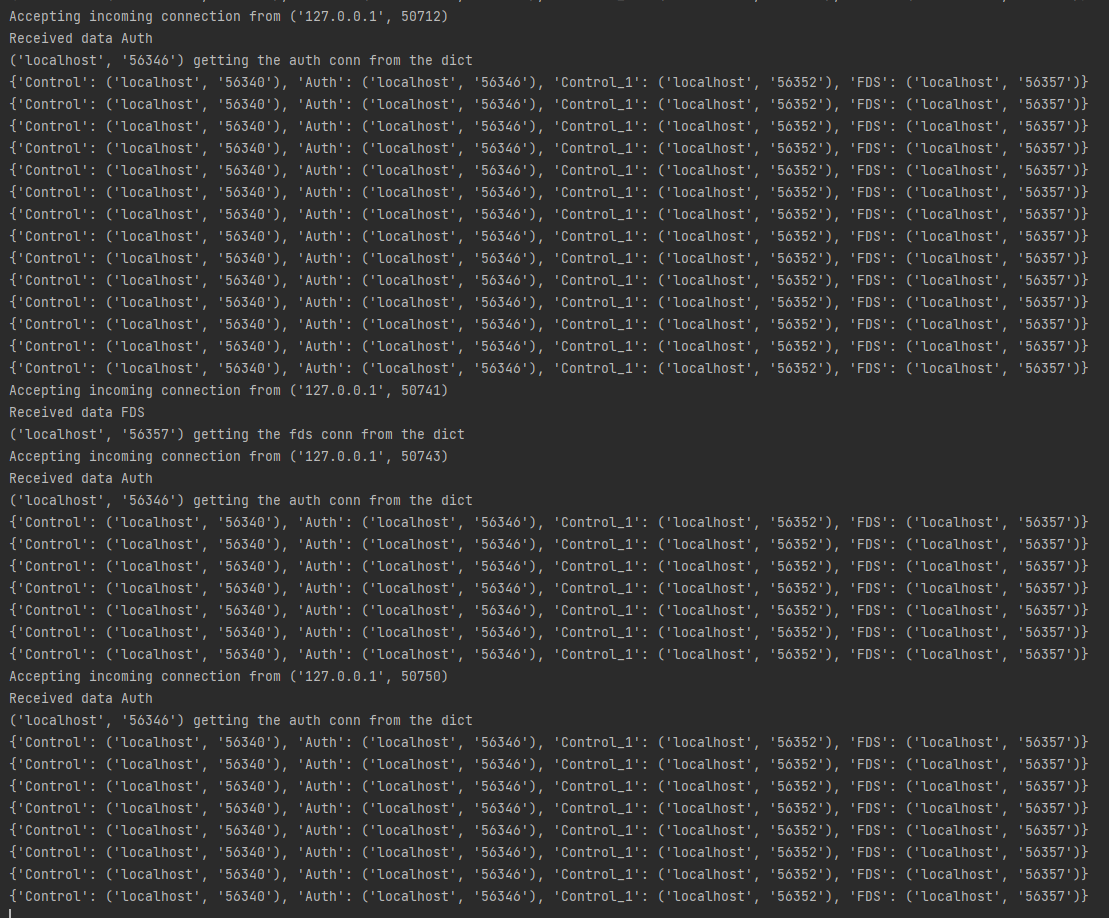
Description automatically generated

I tested if the whole system functions with the user client by starting up the userclient then going through each of the functions it can use to test the system. This also plays the song at the end which was given from the FDS node. This is a run through of all the available commands the user can do on the client and at the end the user playing the requested song.

Text

Description automatically generated

This test shows the connections between the nodes and if it sends back the valid responses. This shows the prime nodes communication between itself and the different nodes and the commands that it is receiving, it also shows that it is sending over the requested connection information for each node that needs that connection when asked.



This shows the authentication node responses and communications between itself and other nodes and the client. This shows the authentication node registering a new user using the username and password they supplied. It also shows the client logging in with a username and password supplied which was checked to see if it valid, then giving a access token to the client. It also shows when the FDS node is trying to check if the clients token is valid, the authentication node checking the token given with the ones stored then validating it and returning the response to the FDS node.

Text

Description automatically generated

This shows the responses and communications to the FDS node. In this it also shows that the node sends over the song for the user client that requested it. This shows the user communication to the FDS node and the validating the token which is given by the client by sending it to the authentication node to be checked. It also shows the client requesting the song list and a song for the client to be sent.

Text

Description automatically generated

The song given from the FDS node saved in the directory of the user client. This shows that the song which is given by the FDS node is downloaded and saved.

Graphical user interface, application, Teams

Description automatically generated

I then tested if the user client was not logged in to test if the system would still allow it to function as if it was logged in. This shows that the validating of the clients token is working correctly.

Text

Description automatically generated

This shows the communications in the FDS node and it notifying the client that they do not have access due to the key which they have being invalid.

Text

Description automatically generated

This shows the authentication node communications and it responding that the token given is not valid.

Text

Description automatically generated

# Outcome of System

The resultant system that I created ended how in it being designed in a way that the core files of the system(server, node, client) can be reused as a base to create different functionality nodes and the control node being setup in a way that is easy to assign a new node to, which would allow it to expand past being limited to what I ended up creating. This design allows the core files to be useful in the future so that the new nodes can use the core node as a base to build on, which will save the repeating code. The client is also setup in a way that will allow more potential inputs to be created and added to the list of available inputs they can use.

The overall outcome of the system was a success in terms of it being a good overall starting point for a system that can be built upon with more functionality nodes that can be easily implemented, and a success overall due to the functionality that has been implemented working. A main object which helped with its success is the modular approach that I used, I did this through making core files that everything would be using and then building upon the core files in the specific nodes file.