100617493 IoT Application

Implementation Logs

19/12/2023

On this day I set the scope of the assignment, I will design my own client-server message system which will allow multiple clients to connect to a central server and send messages.

It will also include server-side error handling for things like clients disconnecting suddenly and for clients wanting to disconnect. Network traffic will also be encrypted using a standard algorithm. Once these are completed, I will look at advancing this scope to include error handling on client side as well as mechanisms to prevent at least 1 kind of attack.

02/01/2024

Today I built my server.py file in which I set the address and port, I also set functions to start the server and use that port to listen for incoming traffic. After this all I had to do was built a function that looped forever that would handle the client, this would include encoding and decoding messages received as well as handling disconnects.

09/01/2024

Today I built the client.py in which I set the local address to connect through on my network and the port I was using. After this I set up my first of two functions which was the message input, this function (while connected) would constantly wait then ask the user for their message to the server with the option to disconnect willingly. Next was my send message function which I would send a message length to get ready for sending the message itself, before this I would encode both into a utf-8 format.

16/01/2024

Today I implemented error handling on the server side, I have added multiple checks within the handle client function that if there is an error thrown there is an exception that occurs and if no connection is reset then an ‘unexcepted disconnect’ message occurs. Furthermore, I have also added the error handle if the client willingly disconnects. If the message received matches the criteria ‘q’ or ‘Q’ then it breaks the connect loop and shows the ‘client disconnected’ message. (Different states based on user input)

18/01/2024

Today I decided which encryption method to use to suffice network security. The standard I chose was AES in which I installed a pip package to my machine and implemented multiple methods both on the client and server side which put the message into a padded format to match the 16 byte-block method used. Once encrypted this is sent and the server side uses a hardcoded AES key to decrypt the message. This was tested using a print statement once the server received the data, and before it went through the decrypt method it showed the encrypted message which sufficed the network security sector.

20/01/2024

On this day I added functions to both my client and server. The client now generates a hash for the encrypted message and combines the two before sending and the server now has a check before decrypting the message, which checks the computed hash and received hash are the same otherwise it throws an exception. This was tested by printing the combined message before sending as well as sending the wrong hash, this provides data integrity and message content verification.

21/01/2024

Today I implemented some client-side features which included handles for when the server is offline as well as when the server crashes unexpectedly. On each loop of trying to send a message if an exception is thrown it is caught into a print statement that says, ‘server error – please restart’. Also, there is another exception handled when the original connection is made that says, ‘server offline’. This provides client-side error handling.

22/01/2024

Today I worked on the last 2 things. The first one being a key exchange using the Diffie Hellman method which also has a failsafe to use the pre shared key instead if it doesn’t work, this ensures security and safety towards the client. The second thing was a state where the server stores data between sessions. This allows clients to request data based on previous sessions.