UNIVERSITY OF CAPE TOWN: COMPUTER SCIENCE

CSC3002F: Networks Assignment 1- 2023

Socket Programming Project

GROUP 22:

1. Mduduzi Ndlovu : NDLMDU011
2. Sphiwe Nyoni : NYNSPH001
3. Matshepo Mathoto : MTHMAT043

DESIGN AND FUNCTIONALITY OF SERVER APPLICATION:

Server implementation

The server is implemented to only accept messages of a certain size from a client at a time. This is the variable buffer size which sets a limit of 4096 bytes. Each file uploaded to the server is stored in a dictionary defined as dictFiles where the filename is used as a key for information on the file. Each file is accompanied by relatively relevant information such as password if the file is protected, visibility (whether the file is visible to all everyone or not), file size, etc. The server implements a variety of methods designed to fulfil user requirements such as uploading and downloading a file. These methods are defined below:

- **get_hex(filename): string**

  - This method uses the hashlib library of python to efficiently hash a file meaning each file will have a unique hash value associate with it.

- **check_sum(server_file, client_file): boolean**

  - This method compared the hash value of two files ie file in the server and file from the client and return true if the files have the same hash value. This method allows us to check the integrity of the files uploaded to or download from either the server or client thus allowing us to be certain that a file did not lose its contents during transmissions

- **receiveFile(conn, filename): void**

  - This method is used for creating a copy of the uploaded file in the storage of the server. It creates a new file of the same type as the one uploaded by the client in the path specific to the server. It tehn receives bytes of messages from the client in contained are the copies of lines or contents of the file being uploaded into the server by the client. These bytes are written in the file created by the server and stored there thus creating a copy of the file of interest in the server's storage.

- **checkAndSend(conn, filename, dictFile): int**

- This method is used in the aid of allowing the client to download a file from the server. Upon receiving a request to download a certain file, file X, the server scans the dictionary containing all the files to check whether the requested file is indeed in the server or not. Sequentially it checks whether the requested file is protected or available to all clients. Only the clients with the encryption key can download a protected file, this method enforces that rule such that protected files can only be downloaded by clients with the correct encryption key, and that open files can be accessed by any client without the requirement of an encryption key. Zero is returned if the encryption key provided is incorrect and one is returned if the opposite is true.

- **sending File(conn, filename): void**

  - This method plays a role in allowing a client to download a file from the server by retrieving the requested file from the server's storage and reading the contents of the file. The contents of the file are sent as bytes until all the bytes are sent.

- **sendingList(conn, dictFile): void**

  - This method allows the client to view all visibly available files in the server that the client is permitted to download. This is possible because the files in the dictionary are stored with information detailing the properties of the file and this includes visibility. A client rendered visible is visible to all clients despite being encrypted or not and a file rendered invisible is shown in the view list despite being an encrypted file.

- **searchFile(filename, dictFile): boolean**

  - Sometimes a client might attempt to download a file that is not present in the server or use the wrong case. This method checks whether a requested file is present in the server despite the case sensitivity of the filename.

- **getFileInfo(filename, dictFiles)[ ]**

  - This method returns a list of information related to a specific file. A standard protocol to save the file info is used. A empty list is return if the file does not exist.

- checkFilename(filename, dictFiles): string

  - This method ensures no two files have the same name by renaming the new file, a standard conversion method is implemented in that the resent file has a number associated with it.

- checkFilename(filename, dictFiles): string

  - This method checks if the requested file in present in the server considering the case sensitivity.

- **main()**

  - The main method is the primary user interface method which creates a socket for the client – server connection specified with a hostname and port number. The server can listen to a maximum of 4 clients at a time. This method makes use of the methods mentioned above to fulfil client requirements

The server offers seven option a user can utilize. The client is allowed to upload a file to and download a file from the server, view public and hidden available files, request information on a particular file, delete a file from the server, and close the connection between itself (the client) and the server. All the files uploaded to the server, which is constantly listening to new connections, are stored in a dictionary for easier access.

ROBUST PROTOCOL IMPLEMENTATION (Stress Tests):

The system does not implement any multithreading meaning our system can perform one query at a time and before performing another query for another client 2. client 1's connection must be closed to be able to establish a connection with client 2. This means that a client must wait for the first client to finish before they could be able to perform their own query. Since simultaneous access is not available a client disconnects after running their query and give space for the client in waiting to connect. The protocol supports uploading files of different types and file sizes with a use of a buffer-size of 4 kilobytes (4096 bytes) per time and transmit the files in seconds or a few minutes.

DESIGN AND FUNCTIONALITY OF INDIVIDUAL CLIENT APPLICATION:

Client 1 (NYNSPH001)

In this client class, the client is permitted to upload and download files to and from the server. They are also allowed to view only publicly visible available files, delete files only if that specific client uploaded the file, request information on a particular file, close the socket connection, and exist. Before the client is able to perform these actions, the client must log into the server using a username and password, this is used to identify the client. If the client wishes to upload a file to the server, they are required to specify whether the file is protected and if the file is publicly visible or hidden. Given that the client wishes to protect the file the client is required to specify the password that can be used to download the file – only clients with this encryption key are allowed to download the file. Given the file being uploaded is not protected, a default password "0000" which won't be required when downloading the file from the server is given to that file and this is done to ensure consistency in keeping information about the file. The information related to a file is made up of visibility status, whether it's protected, encryption key, the hash value of the file, file size, username, and client password. This username and password are important when the client wishes to delete a file from the server since only the client who uploaded the file can delete it from the server, so both the username and password are used to identify the individual client. The client can view files currently in the server that are available for the public to view. This client, however, cannot view files that are hidden. Hidden files cannot be viewed by any client even though that particular client is the one who uploaded the file. The rationale behind this is security. A different client, client X, may get hold of client Y's username and password but this should not allow client X to view every file belonging to client Y. This feature thus provides an extra layer of security. The client implements methods similar to the server and these methods allow the client to perform actions essential to fulfilling requirements and these methods include *download_file* which allows the user to download a file from the server and *check_sum* which allows the client to compare the hash value for it's file and the hash value for the server's file thus conserving the integrity of the transmitted file. The client can perform multiple actions in one implementation.

Client 2 (NDLMDU011)

This client allows the user to connect to the server using a TCP connection which allows sending and receiving messages between them. The server provides a protocol to upload, download, share files and accepts other requests from various clients. This client uses that protocol to send request to Upload a file to the server providing filename, visibility, privacy (encryption key for on selection protected file) and validation measures as info for the file being uploaded. This allows different users to use this client program providing a username and password for their profile and for reference as uploader of the file. The uploading makes use of a hasher method using the contents of the file and returns a string value to compare it with the contents uploaded to the server thus validating that the file was not corrupted in transit. The users can download the files if that file is not protected or the correct decryption of that file was supplied. There's also on display output of the total bytes sent while uploading and total bytes received while downloading to track how the transfer is taking, The

user is also allowed to query about the files available for viewing in the server. Some files are uploaded and set to be visible to certain users who has that specific visibility key. This client allows those users to view and list those files visible to them by that code. This also allows to query for displaying certain information of the file requested by the user. The client also allows the user to remove the file from the server provided that that user profile was the one used to send that file. The client can also flush the previous output in the their use interface using a certain command. Provided by this client is to upload, download, view files, remove them, get their info from the server using the protocol mentioned. Helpful input prompts and feedback is provided in the user interface of this client when running. The user can run once and do unlimited number of queries until opting to close the program. Code was organized and structured using methods, specific variable names.

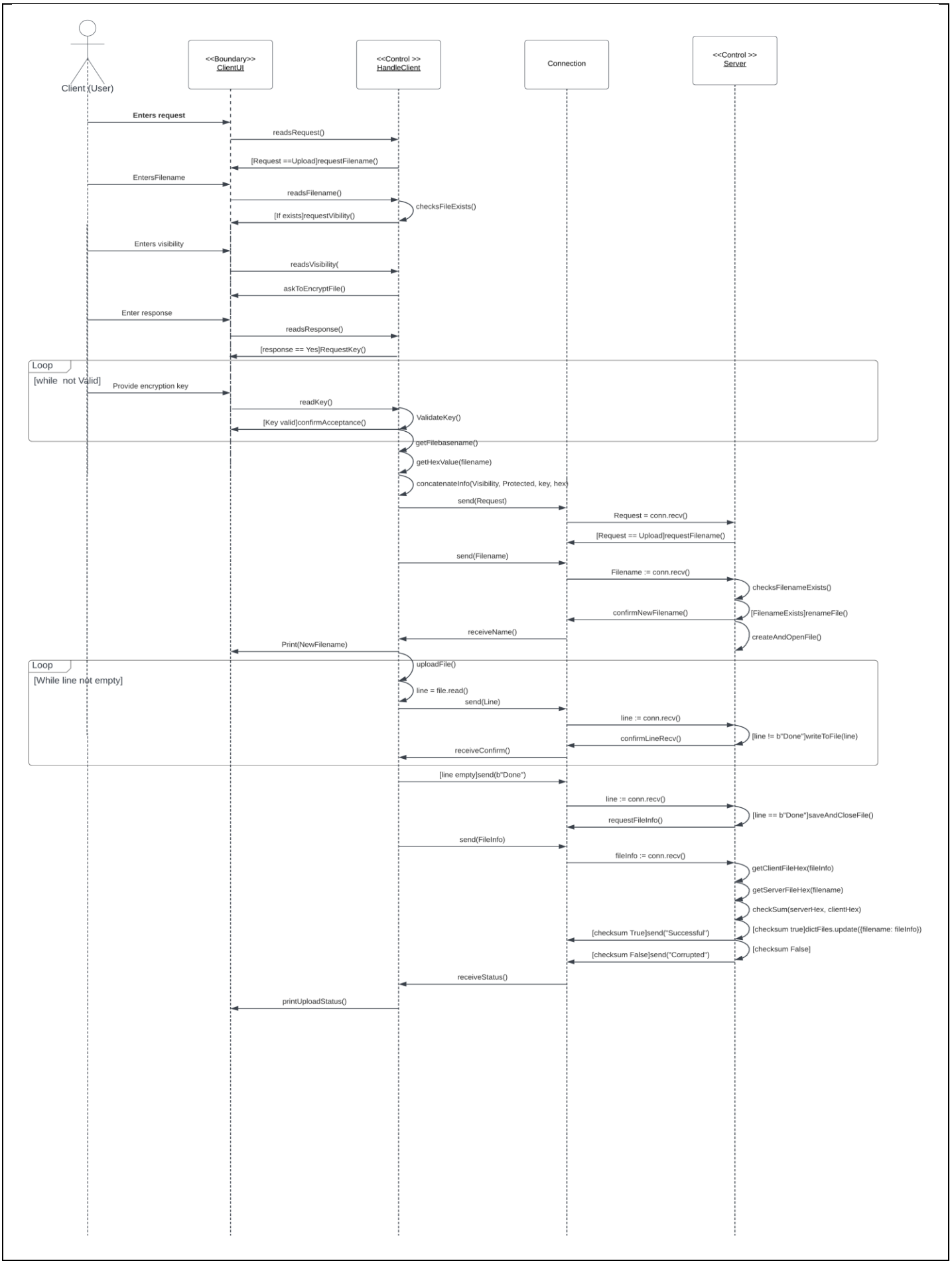
Client 3 (MTHMAT043)

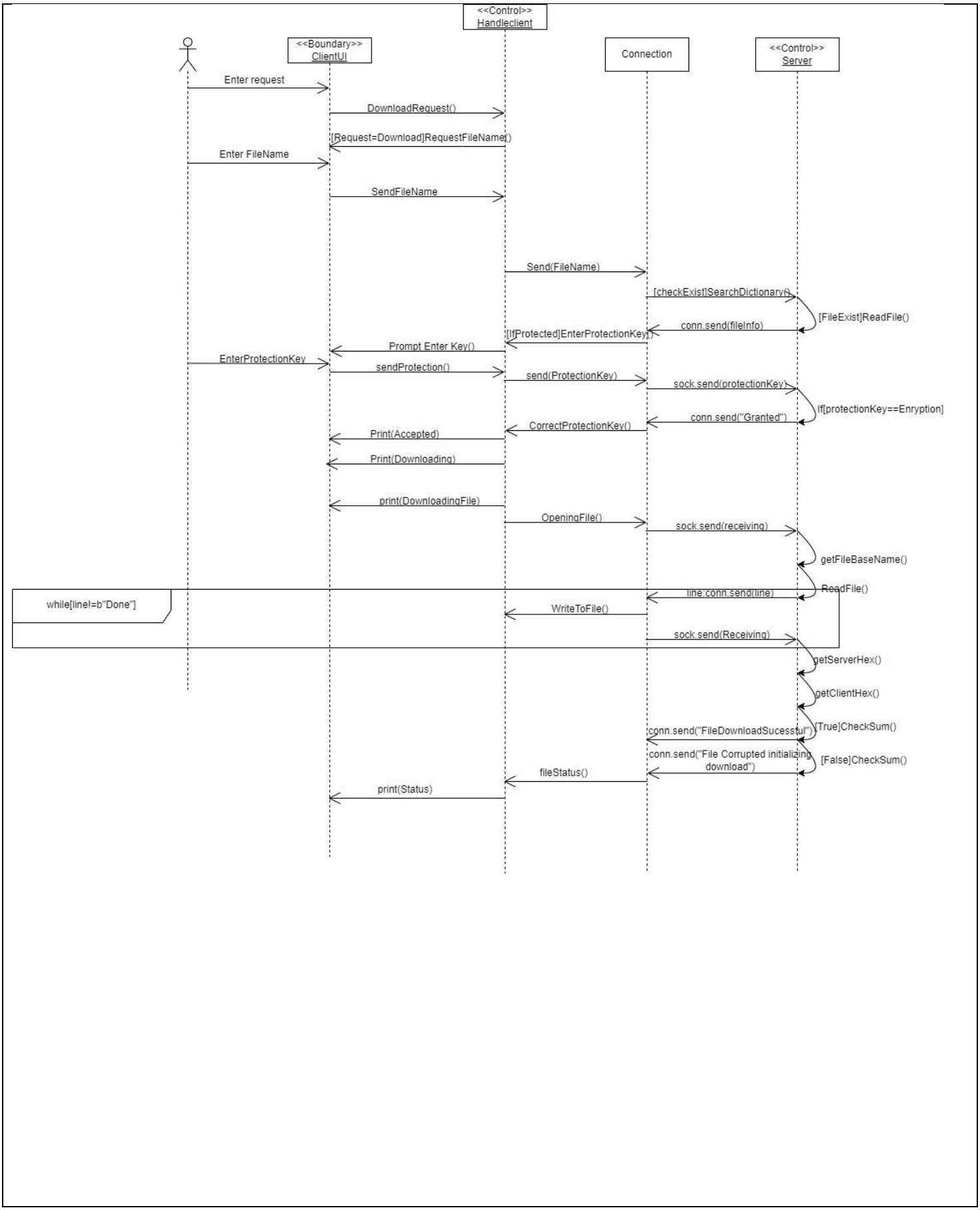
The client class when the user run they must provide port,IPadress and query.If the client did not do this then the default values will be used and the system will ask the user to enter the query. The client will ask be to provide a username and passwords.The client class performs 5 queries depending on which one the client wants to perform. The is upload(U) query that ask the user for filename they want to upload. The method os.path.exist(filename) checks if the file exists in the client's directory, then we get the basename, filesize and the hexValue of the file. For privacy the user can indicate if the file should be encypted. The client will enter the password for encrypting their file and indicate if they want the file to visible to other users. Socket.send(query) this will send the query to the server so the serve knows which query to perfom. Socket.send(basename) will send the file to the server.upload_File(socket,filename,fileSize) this method open the file the user want to upload and send each line to the server and uses the filesize to calculate the percentage of the file transfer. Send the information of the file uploaded to the server to be able to store the file and it's information in the dictionary. Download(D) query, Ask the user for the name of the file they want to download, the receive method will receive the information of the file if the file exists in the server. Before downloading the file the system will check if the file is protected or not, if the file is protected then the client will provide passwords before downloading. DownloadFile()method will download the file and show the percentage of the download until it is complete. Query V which will show a list of all the open files, the hidden files will not be shown at the view query. The request C will close the sockect on the client's side. Query R will delete files that the user uploaded to delete, the username and password will be used to keep track of which file a client can delete. The client will provide the name of the file they want to delete. Query I will provide information about an uploaded file including date of when it was modified and uploaded to the server. Query H will show the hidden files that the client are hidden this will happen if the client provide correct passwords.

LIST OF ADDITIONAL FEATURES:

- Viewing info of requested file: this was added to allow the users to be able to check before downloading if that file is the one they are looking for and whether they have the space to save it on their computer. They are able to view the one who uploaded that file, when it was uploaded, the file size and other useful information.
- Removing a file: this feature is to allow reversal of actions if for instance you uploaded the file by mistake and didn't protect or hide it from other people. This gives a user chance to remove it provided that they are requesting on the same user profile as the one used to upload the file and saved with it on the server. And this is to ensure that not everyone can download files that are not theirs on the server.
- Hiding and viewing files by a certain code or password: this feature is provide a grouping and confidentiality mechanism for files of particular organization, group, team or individuals as their SharePoint to share files between them by uploading all of them with specific visibility or viewing code and also downloads them. Privacy measures are still applicable to encrypt some of those files as well, though they can view them but not allowed to download them without the decryption key.

THE UPLOAD PROCESS SEQUENCE DIAGRAM:

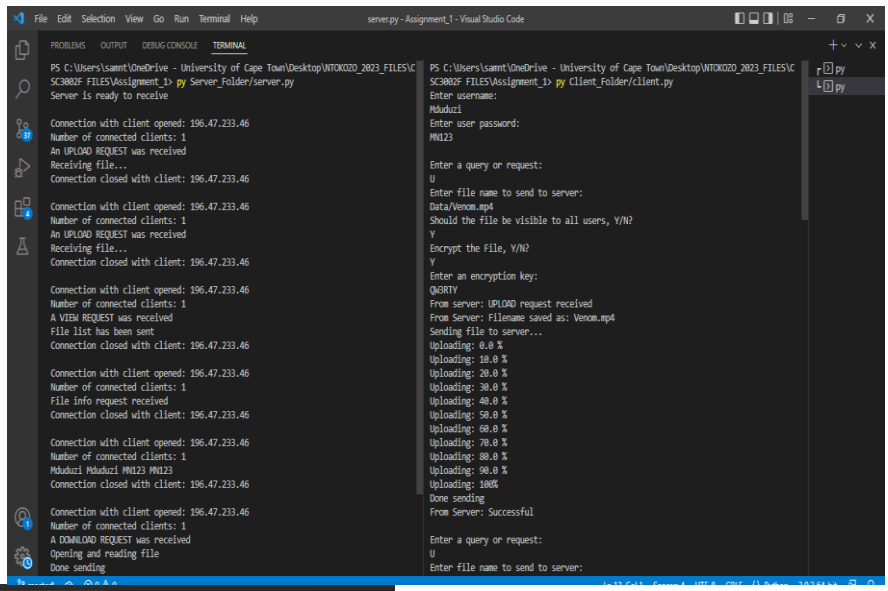


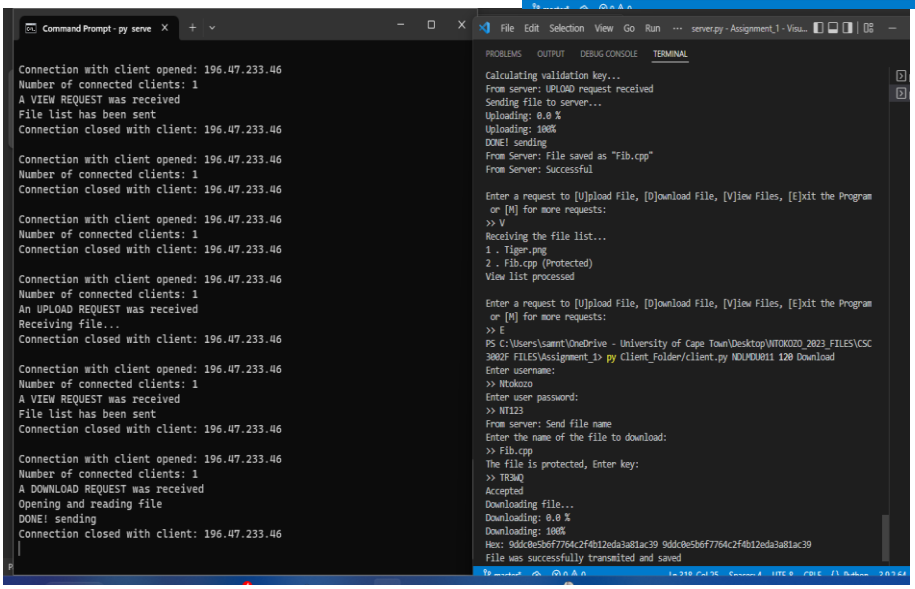


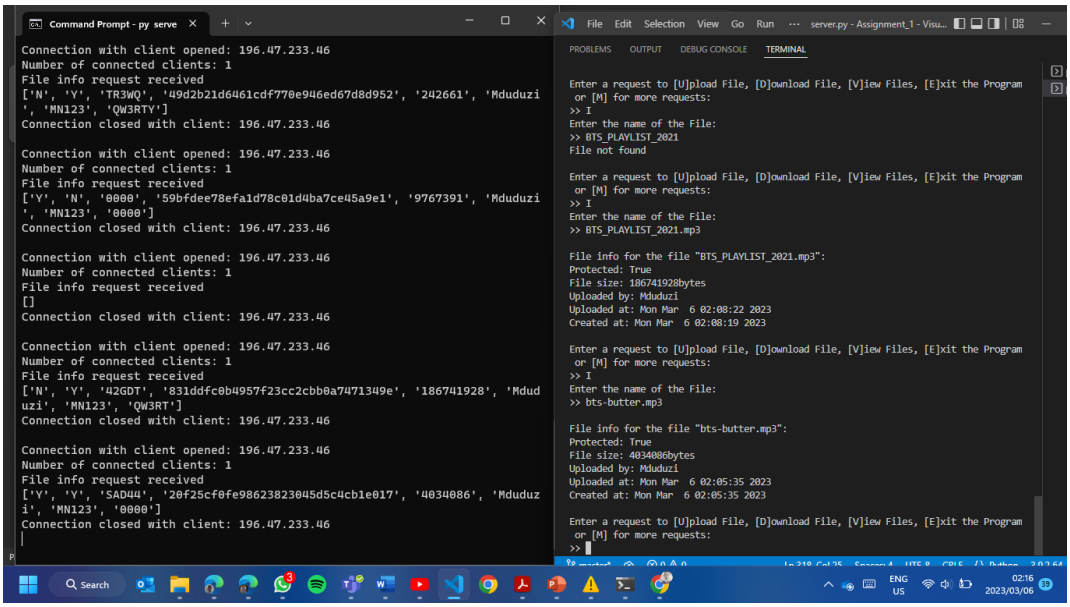
THE DOWNLOAD PROCESS SEQUENCE DIAGRAM:

Screenshot

The screenshot on the right shows a client uploading a mp4 file to the server. Our server can handle large files and can accept a large variety of different file extensions including csv, png, txt, mp4, pdf, etc.







Note: Some messages were modified on the final files uploaded and so output may be slightly different to ones shown and those when the server-client protocol is tested.