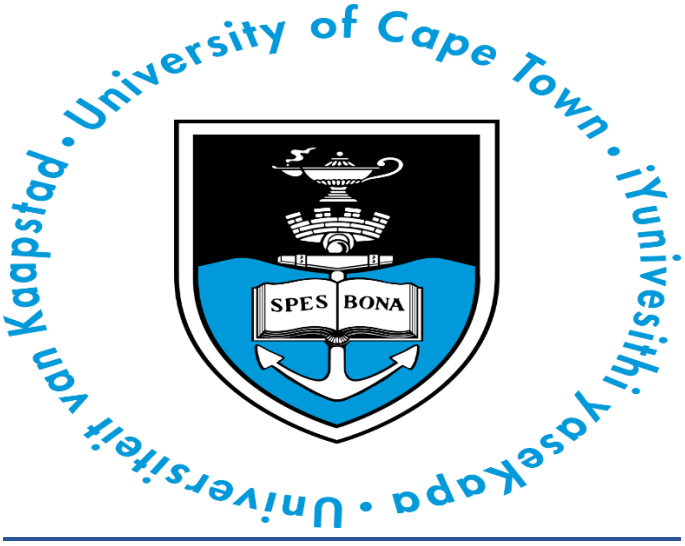


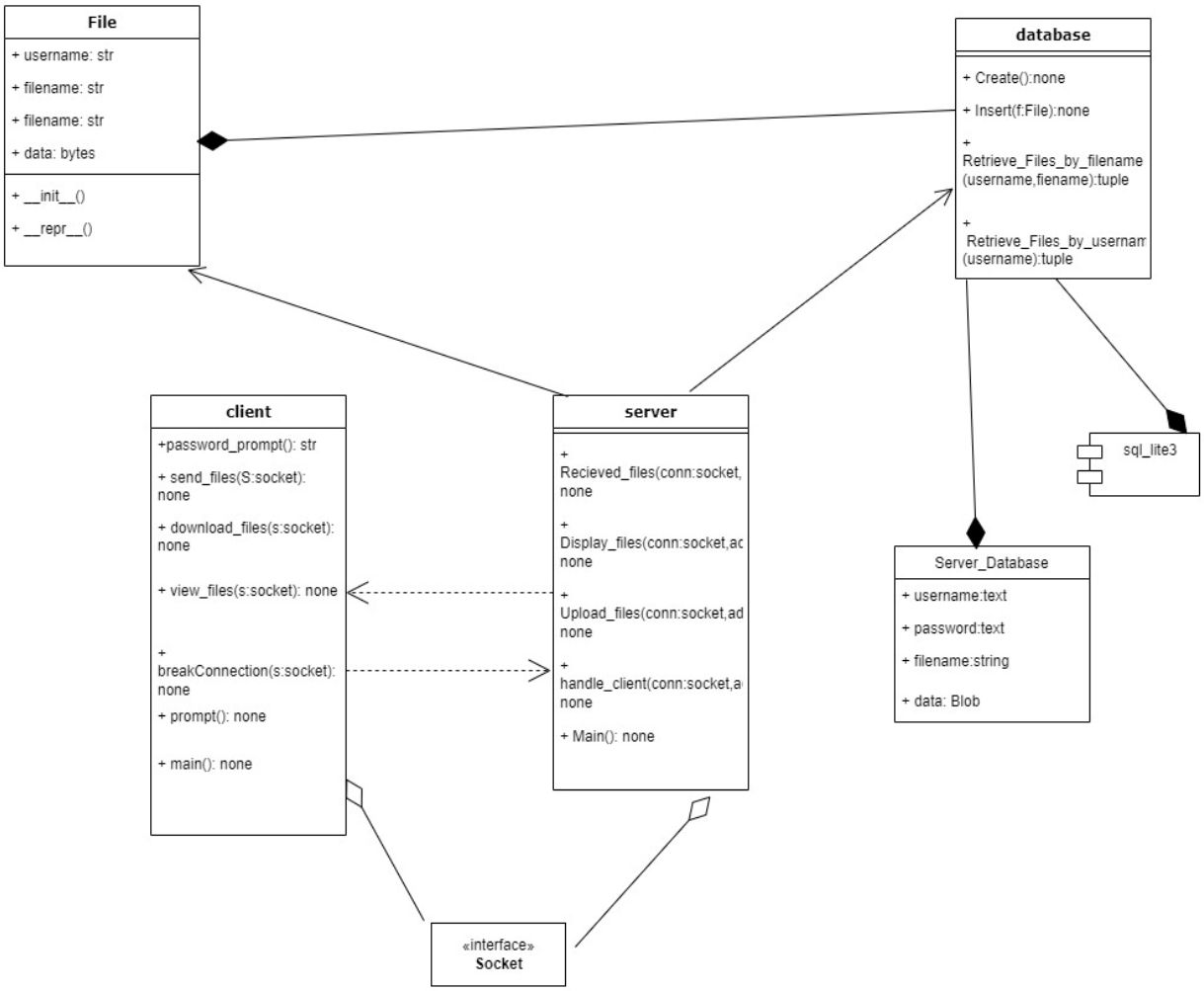
CSC3002f– Networks Assignment 1–REPORT
2023 Socket programming project

Sydney muganda MGNSYD001

I. Introduction:

The assignment was on building a network application. Here I used the basics of protocol design and socket programming for TCP connections in Python to create a socket, bind it to a specific address and port, as well as send and receive messages and files from a server to a client and vice versa . I then developed a client-server application to demonstrate these functionalities. This document describes the overall features , functionalities, protocol design and class descriptions .

II. Class outline and descriptions



As seen above, the client and server programs form the basis of the application. When the main method of each program(server first then client)is run, the **client** will connect to the server then send out a prompt to the user.

From the message prompt the user can determine what they would like to do whether to send, receive or view files. The subsequent function will then be used for each action.

From the **server** side the program will run then create a database to store all files sent to it from clients and use the functions to handle client requests , whether uploading, downloading or viewing

The server also allows multithreading to enable multiple clients to connect at once using the handle_client method.

The database is a script with functions that allow thread safe access to the Server_files database table. It will safely create ,update , insert and retrieve records from the database table using its function.

The FileIO class is used to create an object instance if type File which stores the details of a file from username, filename , password and file data.

III. Description of system functionality and features:

The following are the in depth walk-throughs of system features and functionalities of each function of the server-client application:

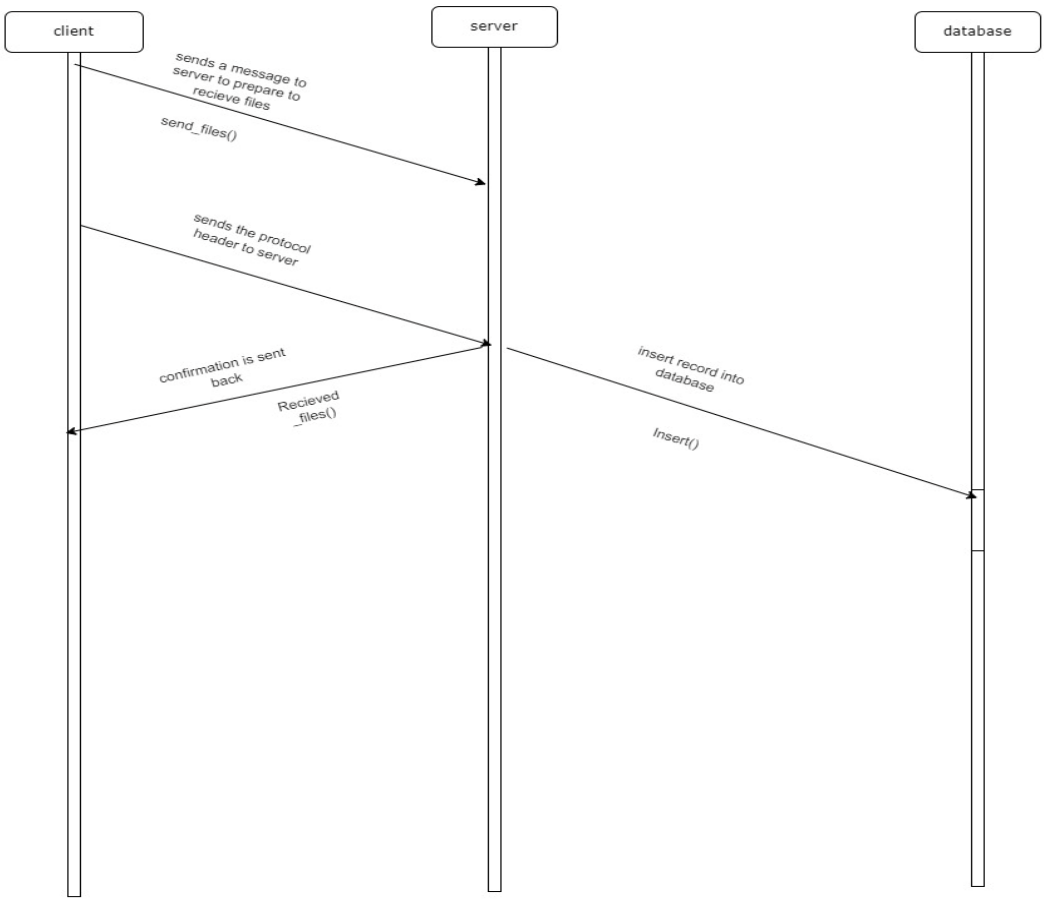
First and foremost, the server program is started , when it starts it creates a database table called Server_fies if said Table does not exist, if it does exist it will simply send a feedback that the database is present and ready for use. It will then listen to and wait for client connections.

then the client program is started , here the program will ask the client to enter their username then allows the client to connect to the server with a given port and local host. The client is then prompted to select between 4 options whether to send ,receive ,display files or disconnect. A running loop will retain connection until the client manually disconnects from server.

This makes the initial action message sent will always be correct.

The server then receives the action message and does one of the following:

1.Sending files:



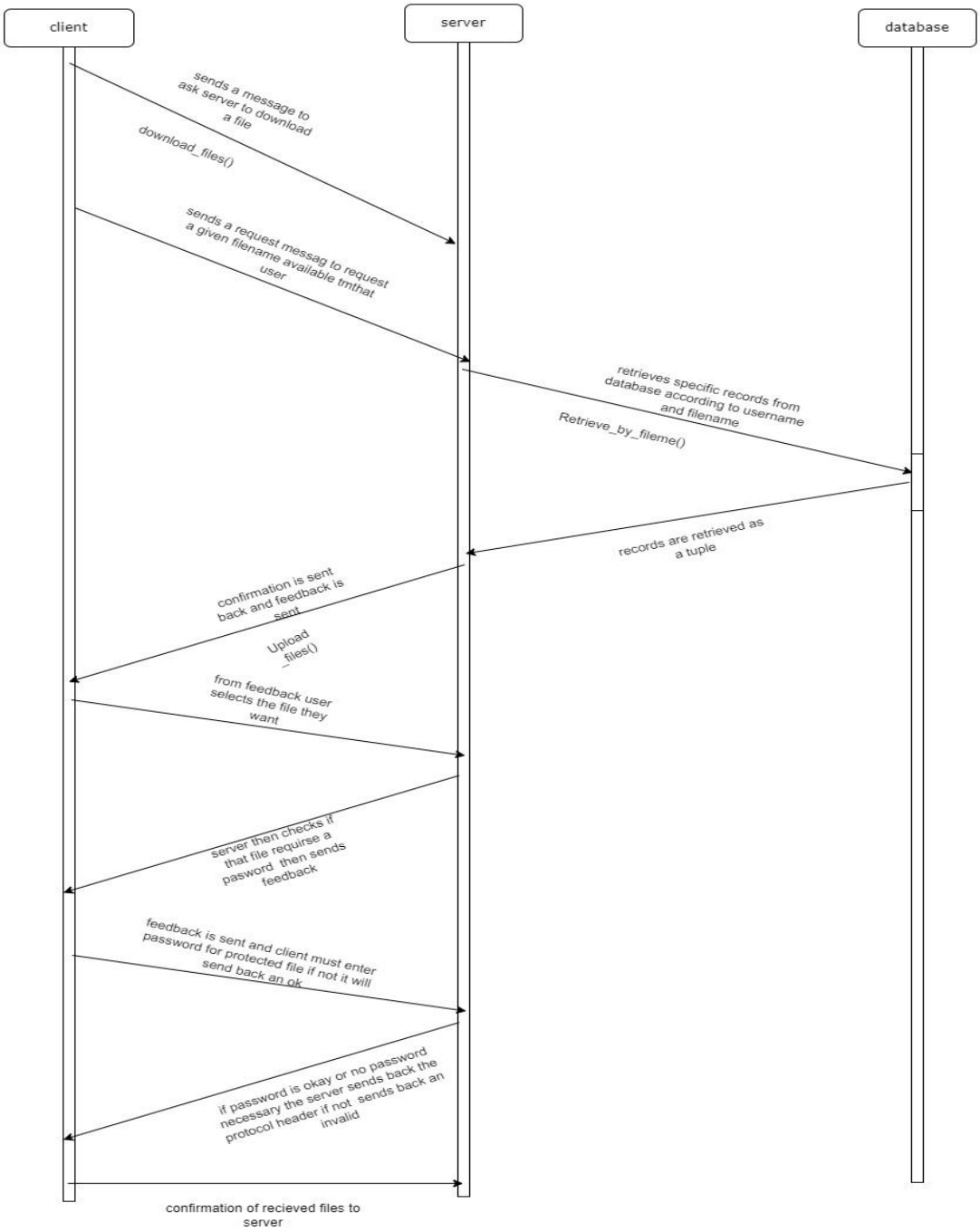
When sending files the client first sends a message to the server to prepare the server to receive files. This message is the action message and is prompted from the start of the server application. The server then receives the action request and prepares to receive files by calling theRecieve_files() method.

The client then is simultaneously prompted to enter the name of the file he/she would like to send and is then sends a protocol header .

Protocol Header design: the protocol header here contains the filename, the destination username(the user the file is being sent to), password (if protected if not “none”) and file data . all encoded as bytes.

The server then receives the header processes it then stores the files information in the database then finally sends back confirmation that it has received the file.

2.Recieving files:


client
server
database
sends a message to ask server to download a file
download_files()
sends a request messag to request a given filename available tmthat user
retrieves specific records from database according to username and filename
Retrieve_by_fileme()
records are retrieved as a tuple
confirmation is sent back and feedback is sent
Upload _files()
from feedback user selects the file they want
server then checks if that file requirse a pasword then sends feedback
feedback is sent and client must enter password for protected file if not it will send back an ok
if password is okay or no password necessary the server sends back the protocol header if not sends back an invalid
confirmation of recieved files to server

When Receiving files, the client first sends a message to the server to prepare the server to upload files. This message is the action message and is prompted from the start of the server application. The server then receives the action request and prepares to upload files to client by calling the Upload_files() method.

The client then is simultaneously prompted to enter the name of the file he/she would like to download and is then sends it to the server .

The server receives both the client's username and the requested filename and then queries this in the database using the Retrieve_file_by_username().

The database returns a list of records corresponding to the filename and username(if file is unprotected it means it can be accessed by everyone) Then sends back this list to the client as feedback.

The client receives this list then chooses the desired option. And sends it back to the server.

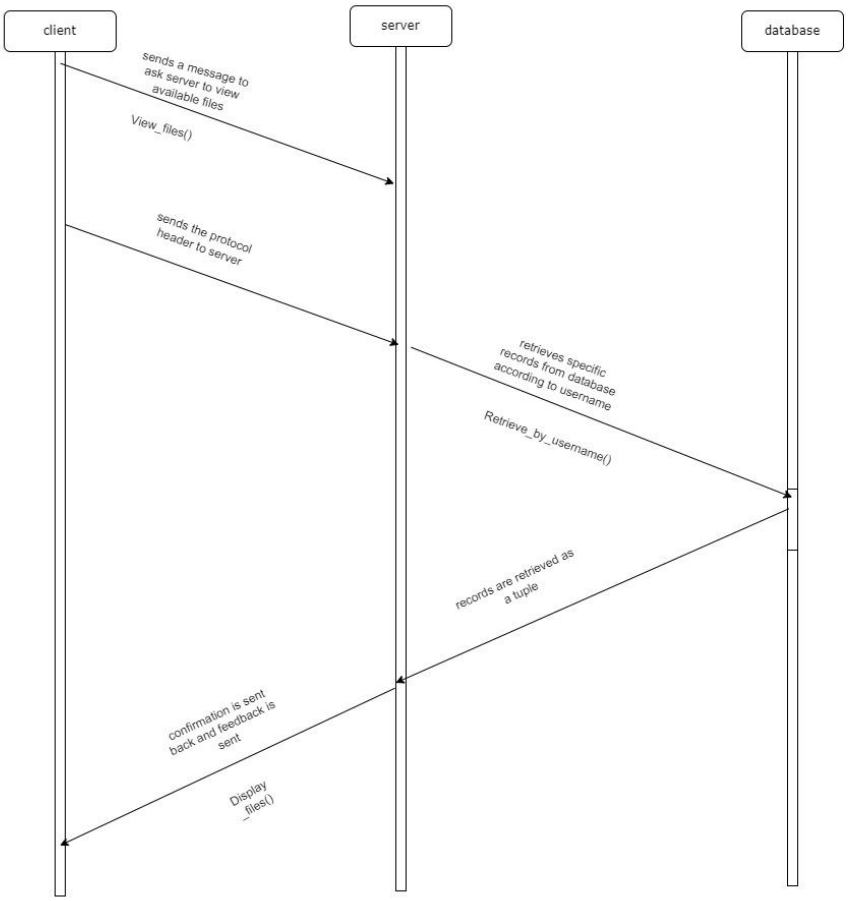
The server picks that file, checks if it needs a password and sends it back to the user. If it needs a password the user will be prompted to enter a password. The password will then be sent to the server . if not a ok reply will be sent back to the server then the action terminated

If the password entered is correct the file will be accessed, and a protocol header will be sent back to the client from the server.

Protocol Header design: the protocol header here contains the filename, file-size, and file data . all encoded as bytes.

The client then receives the header, processes it then stores the downloaded files in the downloaded files then finally sends back confirmation that it has received the file.

3.Viewing files:



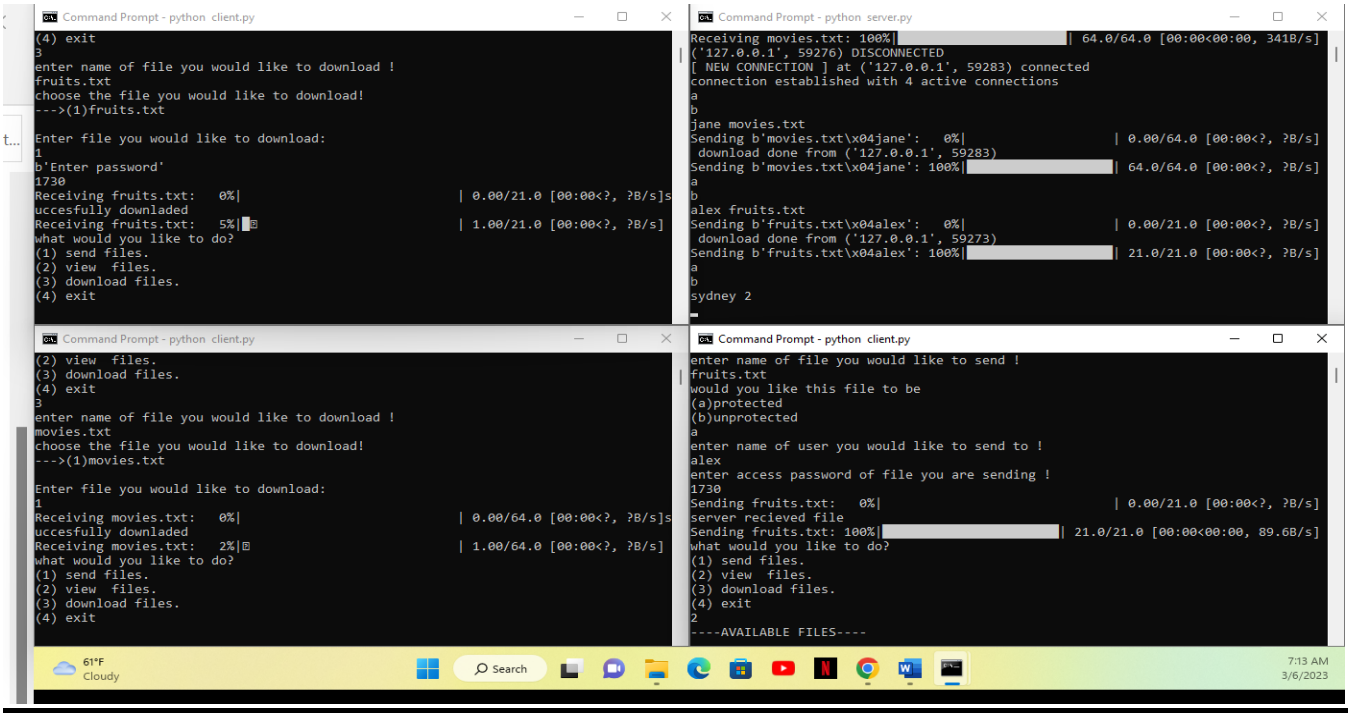
When displaying available files, the client first sends a message to the server to request the server to display files. This message is the action message and is prompted from the start of the server application. The server then receives the action request and prepares to display files by calling the Display_files() method.

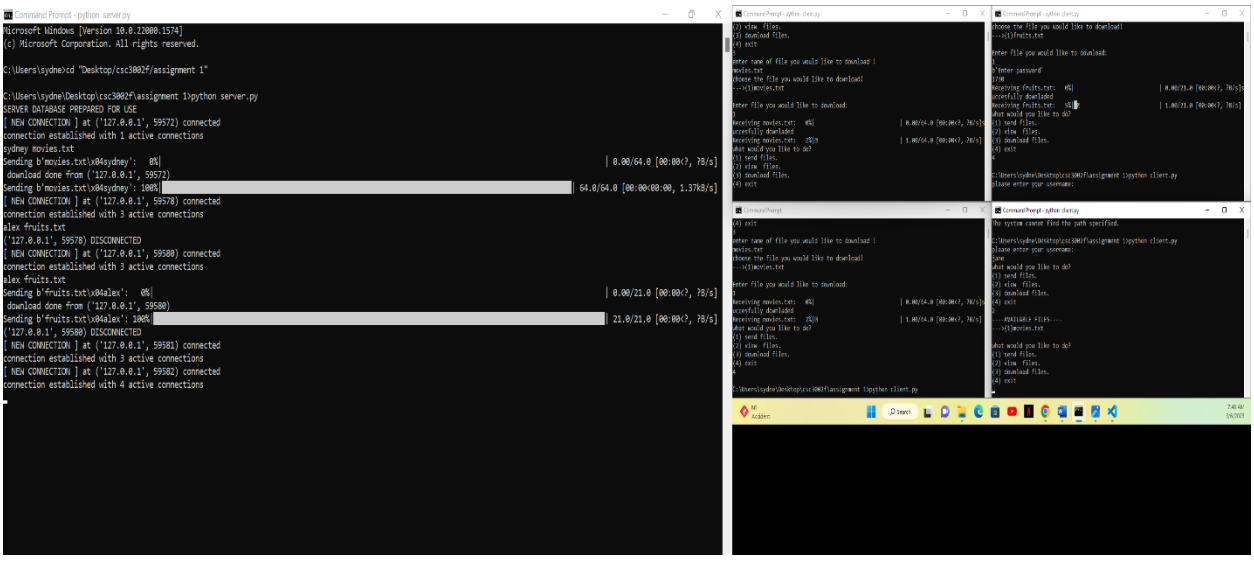
The client then sends a protocol header .

Protocol Header design: the protocol header here contains the client's username encoded as bytes.

The server then receives the header processes it then requests the given file from the database using the Retrieve_files_by_username() method the files information in the database then finally retrieved as a list of records then sent back to the client to view.

IV. SCREENSHOTS OF APPLICATION





- *Additional/creative features:* a progress bar was added to check the progress of downloading and uploading on both the server and client side, this makes for the creative features.

- *Stress testing:*
  I tested the application with multiple clients up to 5-7 and all were able to connect to the server
  The database was made thread safe so no two or more threads can access it at a time.
  Most scenarios were tested.
  This application only works with text files.
  Sample text files were given.
  The application did well with text files up to 4mb large, but larger files proved difficult.

- *Note to marker:*

-Progress bar only works if you install the tqdm library using the pip install tqdm command in the terminal
-I could not test the application on local host but for connection across multiple computers hard coding during presentation would be required
-the downloaded_files folded must be present in order to store all downloaded files onto that folder
-all files meant to be sent must be put in the same location as client
-the application only works with text files
- I did the assignment by myself , since there was a breakdown in communication with my group and I was left to do a majority of the coding for the server, client and writeup by myself I therefore request an individual presentation

V. **Conclusion:**

The application successfully sent files from client to server . Large files proved challenging as sending files across a network but overall it was a beneficial learning curve