PROJECT REPORT ON

THE MEDIA STREAM

SUBMITTED IN PARTIAL FULFILLMENT OF THE AWARD OF

CBSE Senior Secondary Certificate

2019-2020

DONE BY

NAME: ROHAN ANIL KUMAR

REGNO: 24634304

UNDER THE GUIDANCE OF

Mr. SHAJI.M, PGT Computer Science

KENDRIYA VIDYALAYA

EZHIMALA

KANNUR (DIST), KERALA

KENDRIYA VIDYALAYA EZHIMALA

CERTIFICATE

Certify that the project entitled ***“THE MEDIA STREAM “*** *is a software created by*

Name: ROHAN ANIL KUMAR

Reg. No: 24634304

During the period of 2019 – 2020 of his/her study in this school under the guidance of Mr. M.SHAJI, PGT Computer Science   in partial fulfilment of the requirement for the CBSE Senior Secondary Certificate.

Submitted for practical Examination held on ……………………...

Project Guide

Teacher in-charge

External Examiner

Principal

DECLARATION

I ROHAN ANIL KUMAR, **reg. no. 24634304 hereby declare that the project entitled "*THE MEDIA STREAM*" is the result of original work done by me during the period of study in class XII of computer Science, *KENDRIYA VIDYALAYA, EZHIMALA* under the supervision and guidance of *Mr. M.Shaji*, PGT Computer Science.**

This project report is submitted in fulfilment of the requirement for the award of CBSE Senior Secondary Certificate.

Place: **Signature:**

Date:

ACKNOWLEDGEMENT

I express my sincere thanks to **Mr.Muraleedharan T** principal of our vidyalaya, who has created conductive atmosphere to finish my project in time.

A sincere and true word of thanks to Mr. SHAJI.M,

PGT computer science for his valuable suggestion, guidance and encouragement.

I thank all my sincere friends who helped me for the successful completion of the project.

Last but not the least I would like to thank the almighty and my parents for showering their blessing on me.

ABSTRACT

This is a project report on a software that helps in setting up a client-server file sharing via the Local Area Network. It uses TCP protocol to ensure that no files get corrupted during the process. The main objective of this project is resource sharing. It is written in python and hence the simplicity of the programming language can be seen reflecting off of it. This project report serves as a catalogue and the complete breakdown of the code to make it seem as simple as possible.

Python:

Python is an easy to use, object oriented, high level programming language. The programmer can easily read and interpret the code due to its simplicity.

The program is interpreted using the python interpreter.

INDEX

INTRODUCTION

Think of a system where we have a centralized bulky server with all our personal files. Now think of a portable machine like a laptop that needs to access these files. Traditionally we would use a USB flash drive to transfer files. This is tedious and time consuming. We solved the problem by making the server accessible to all the portable devices through a local area network (wired/wireless). Now everyone can access these files hassle free. Many devices can be connected to the server at the same time and hence none of your work is hindered.

This software that we've created makes use of the local area network to send and receive large files at 100MB/s. This is an easy and much faster approach than using a USB flash drive to transfer files.

If the LAN is large enough, the user can access the files on the server which is far from him.

All of these sums up to unhindered workload and easier accessibility of files.

One of the main features of this software is that it reduces data redundancy. One file can be stored in the server and other devices can request the file whenever needed rather than storing numerous copies of files on different machines.

If WLAN is used, it's almost like portable devices having petabytes of data that can be read and written extremely fast.

SYSTEM CONFIGURATION

The configurations of the development machine:

Hardware:

- Intel Core i7 processor
- 1080p display
- 8GB DDR4 RAM
- 1 TB hard drive
- Qualcomm Wireless Network card

Software:

- Windows 10
- Python 3.6.5 installed
- Visual Studio 2017

However,

MININUM SYSTEM REQUIREMENTS:

HARDWARE:

- Any Intel/AMD processor, even ATMEL will run
- 50MB KB RAM
- 10MB hard drive space
- Wireless/Wired Network card and network installation

SOFTWARE:

- Any windows/Linux OS
- Python 3 or above installed

DEVELOPMENT MACHINE

WINDOWS 10:

Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems. It is the successor to Windows 8.1, and was released to manufacturing on July 15, 2015, and broadly released for retail sale on July 29, 2015. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users. Mainstream builds of Windows 10 are labelled version YYMM with YY representing the year and MM representing the month of release. For example, Version 1809 for September 2018. There are additional test builds of Windows 10 available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

One of Windows 10's most notable features is support for universal apps, an expansion of the Metro-style apps first introduced in Windows 8. Universal apps can be designed to run across multiple Microsoft product families with nearly identical code—including PCs, tablets, smartphones, embedded systems, Xbox One, Surface Hub and Mixed Reality. The Windows user interface was revised to handle transitions between a mouse-oriented interface and a touchscreen-optimized interface based on available input devices—particularlyon 2-in-1 PCs, both interfaces include an updated Start

menu which incorporates elements of Windows 7's traditional Start menu with the tiles of Windows 8. Windows 10 also introduced the Microsoft Edge web browser, a virtual desktop system, a window and desktop management feature called Task View, support for fingerprint and face recognition login, new security features for enterprise environments, and DirectX 12.

PYTHON:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until stepping down as leader in July 2018.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

MODULES USED

Modules are small bundles of code that are written by programming wizards that help other programmers to simplify and optimize their code.

OS:

This package in python is used to interact with the command line of the operating system. As python did not provide an easy way to refresh the console window, we had to run the CLS command in command prompt.

This command can be run using

os.system('cls')

The os.system() function takes an argument which is passed on to the command prompt.

Tkinter:

This package provides all the necessary elements for making a GUI based application on python. It is an inbuilt one so it doesn't need any tiring installation process.

SOCKET:

This module is exclusively used for socket programming in python. This module enables us to connect and share data between computers in the same network.

TIME:

This module provides elements to stop and resume execution. This becomes crutial for network software.

MULTIPROCESSING:

This module helps to run different python on the same project parallel. This is used by the server to handle different clients at the same time.

SHUTIL:

Similar to OS module but with more functionalities.

The best part about these modules are that each and every one of them comes inbuilt with the python standard library. Hence there is no tedious task to download any packages for running the software.

The Program

Server.py:

```
import socket

import os

from multiprocessing.dummy import Pool

import shutil


class Server:

    pwd="0000"

    s=socket.socket()

    clients=[]

    authorizedclients=[]

    busyclients=[]

    new_Client=None

    def __init__(self,root,pwd='0000'):

        self.pwd=pwd

        port=12346

        self.serverroot=root

        hn=socket.gethostname()

        #self.s.bind((socket.gethostbyname(hn),port))

        self.s.bind(('127.0.0.1',port))

        self.s.listen(5)


        print("Server running at",socket.gethostbyname(hn),":",port)


    def getClient(self):

        global new_Client
    while True:
        c=self.s.accept()[0]
        self.clients.append(c)
        self.new_Client=c
        print("Connected")
        self.authorize(c)
        if(c in self.authorizedclients):
            print("Successfully authorized")
            p=Pool(processes=1)
            p.apply_async(self.getRequest)

            #self.getRequest()
        else:
            print("Client removed")

def authorize(self,c):
    try:
        while c not in self.authorizedclients:
            password=c.recv(100)
            print(password)
            if(password==self.pwd.encode()):
                self.authorizedclients.append(c)
                print("Done")
                c.send(b"Authorized $ "+(self.serverroot).encode())
            else:
                c.send(b"Nope")
    except:
        try:
            self.clients.remove(c)
            new_client=None
        except:
            pass
def waitbusy(self,c):
    while c in self.busyclients:
        pass

def deleteFile(self,c,path):
    os.remove(path)
def deleteFolder(self,c,path):
    shutil.rmtree(path)
def Rename(self,c,oldpath,newpath):
    os.rename(oldpath,newpath)
def returnError(self,c,Err):
    c.send("Error")

def createFile(self,c,path):
    dir='\\'.join(path.split('\\')[:-1])
    fName=path.split('\\')[-1]
    print(f"Creating File {fName} on {dir}")
    f=open(dir+'\\'+fName,'w+')
    f.close()

def createDir(self,c,path):
    dircheck=path.split('\\')[0]+"\\"
    dir=path.split('\\')[1:]
    print(dir,dircheck,path)
    for i in dir:
        if(i!=''):
            if(i not in os.listdir(dircheck) ):
                print("creating dir",i,'on',dircheck)
                os.mkdir(dircheck+"\\"+i)
            dircheck+=(i)
            dircheck+='\\'

def getRequest(self):
    c=self.new_Client
    while True:
        self.waitbusy(c)
        try:
            byte=c.recv(1)
            request=b''
            while byte!=b"^":
                request+= byte
                #print(byte)
                byte=c.recv(1)
        except:
            self.clients.remove(c)
            break
        #print("waiting for request")
        print(request.decode())
        if(b'Get file' in request):
            print((request.partition(b"$")[2]))
            self.sendFile(c,(request.partition(b"$")[2]).decode())
```

```
          if(b'Get Dir' in request):
              self.sendDirectoryList(c,(request.partition(b":")[2]).decode())
          if(b'Create File' in request):
              print("File create request",(request.split(b'$')[1]).decode())
              self.createFile(c,(request.split(b'$')[1]).decode())
          if(b'Create Dir' in request):
              print("Dir create request",(request.split(b'$')[1]).decode())
              self.createDir(c,(request.split(b'$')[1]).decode())
          if(b'Sending File' in request):
              RecieveData=request.split(b'$')
              print("Recieving
File",(request.split(b'$')[1]).decode(),'\nCommand=',RecieveData)

self.GetFile(c,(RecieveData[1]).decode(),RecieveData[2].decode(),int(RecieveData[
3].decode()))
          if(b'Sending Folder' in request):
              RecieveData=request.split(b'$')
              print("Recieving Folder",(request.split(b'$')[1]).decode())

self.GetFolder(c,(RecieveData[1]).decode(),RecieveData[2].decode(),RecieveData[
3].decode(),RecieveData[4].decode())
          if(b'Delete File' in request):
              print("Deleting File",(request.split(b'$')[1]).decode())
              self.deleteFile(c,(request.split(b'$')[1]).decode())
          if(b'Delete Folder' in request):
              print("Deleting Folder",(request.split(b'$')[1]).decode())
              self.deleteFolder(c,(request.split(b'$')[1]).decode())
          if(b'Get Folder' in request):
              print("Folder request",(request.split(b'$')[1]).decode())
```

```
                self.sendFolder(c,(request.split(b'$')[1]).decode())
            if(b'Get File Size' in request):
                print("FileSize request",(request.split(b'$')[1]).decode())
                self.sendFileSize(c,(request.split(b'$')[1]).decode())
            if(b'Rename' in request):
                print("Rename
request",(request.split(b'$')[1]).decode(),"to",(request.split(b'$')[2]).decode())

self.Rename(c,(request.split(b'$')[1]).decode(),(request.split(b'$')[2]).decode())
            if(b'Shutdown' in request):
                self.clients.remove(c)
                break
    def fileLayeredSearch(self,path):
        #print("f "+path)
        try:
            if(path[-1]!="\\"):
                path=path.rstrip()+"\\"
            files=os.listdir(path)
            temp=[]
            for i in files: temp.append(path+"\\"+i)
            files=temp
            #print("asdfad")
        except:
            return
        for i in files:
            if os.path.isdir(i):
                morefiles=self.fileLayeredSearch(i)
                if(morefiles!=None):
```

```
            files.extend(morefiles)
    return files


def sendErr(self,c,ErrMsg):
    c.send((f"^{ErrMsg}^").encode())



def sendFolder(self,c,path):
    sep_string=";*&#%@"
    self.waitbusy(c)
    try:
        print(path)
        FileNames=self.fileLayeredSearch(path)
        if(FileNames==None):
            self.sendErr(c,"Directory not found")
            return
        self.sendErr(c,"None")
        FileSizes={}
        FileSendStr='^'
        for i in FileNames:
            print("packing",i)
            if(os.path.isfile(i)):
                FileSizes[i]=os.path.getsize(i)
        for i in FileNames:
            if(os.path.isfile(i)):
                FileSendStr+=(i+"$"+str(FileSizes[i])+sep_string)
            else:
                FileSendStr+=(i+sep_string)
            FileSendStr+="^"
            c.send(FileSendStr.encode())
            print("File metadata sent")
        except(FileNotFoundError):
            self.sendErr(c,"FileNotFound")

    def sendDirectoryList(self,c,DPath):
        self.waitbusy(c)
        self.busyclients.append(c)

        print("Requestind Dir")
        try:
            dirs=os.listdir(DPath)
            self.sendErr(c,"None")
            c.send(b"^")
            for i in dirs:
                if(os.path.isfile(DPath+"\\"+i)):
                    c.send((i+"f;").encode())
                else:
                    c.send((i+"d;").encode())



            c.send(b"^")


            self.busyclients.remove(c)
        except:
            self.sendErr(c,"FileNotFound")
            print("SomeErr")
```

```
def sendFileSize(self,c,path):
    self.waitbusy(c)
    self.busyclients.append(c)
    try:
        size=str(os.path.getsize(path))
        self.sendErr(c,"None")
        stream=("^"+size+"^").encode()
        c.send(stream)
    except:
        self.sendErr(c,"FileNotFound")
    self.busyclients.remove(c)

def sendFile(self,c,path):
    self.waitbusy(c)
    self.busyclients.append(c)
    try:
        chunk=1024*1024*1024
        print("sending file")
        FileName=path.replace("\\\\","\\")
        FileSize=os.path.getsize(path.replace("\\\\","\\"))
        self.sendErr(c,"None")
        print(f"File size ",FileSize)
        f=open(FileName,'rb')
        x=True
        ByteCount=0
        while FileSize-ByteCount>=chunk:
            data=f.read(chunk)
            print("Loop")
            c.send(data)
            ByteCount+=chunk
        if(FileSize-ByteCount>0):
            c.send(f.read(FileSize-ByteCount))
        print("File sent")
    except:
        self.sendErr(c,"FileNotFound")
    self.busyclients.remove(c)

def GetFile(self,c,path,fname,size):
    print("Geting file start")
    self.waitbusy(c)

    self.busyclients.append(c)

    chunks=1024*1024
    if(not os.path.exists(path.replace(fname,""))):
        os.makedirs(path.replace(fname,""))
    print("Writing File",path+"\\"+fname)
    with open(path+"\\"+fname,"w+") as f:
        pass
    if(size>0):
        recvsize=0
        with open(path+"\\"+fname,"wb") as f:

            while size-recvsize>0:
                if(size-recvsize<chunks):
                    data=c.recv(size-recvsize)
                    f.write(data)
                    recvsize+=len(data)
                else:
                    data=c.recv(chunks)
                    f.write(data)
                    recvsize+=len(data)
            if(size-recvsize>0):
                f.write(c.recv(size-recvsize))

    self.busyclients.remove(c)
    print("Getting file stop")

def GetFolder(self,c,path,FolderName,DirCode,FileCode):

    self.waitbusy(c)

    Dirs=[]
    Files={}
    for i in DirCode.split("|"):
        if(i!=""):
            Dirs.append(i)
            #print("Dir",i)
    for i in FileCode.split("|"):
        if(i!=""):
            Files[i.split(";")[0]]=i.split(";")[1]
            #print("File",i)
    for i in Dirs:
        if(not os.path.exists(i)):
            os.makedirs(i)

        for i in Files:
            self.waitbusy(c)
            c.send(("^"+i+"^").encode())
        c.send("^$$$$$$$$^".encode())




#p=Pool(processes=1)
#p.apply_async(getClient)
s=Server(input('Server Root Directory: '),input("Server Password: "))
s.getClient()
#print(s.FileLayeredSearch("E:\\"))
```

Client.py:

```
import socket
import time
import os

class Client():


    s=socket.socket()

    def __init__(self,ip,port):
```

```
    self.s.connect((ip,port))
    print("Connected")




def authorize(self,password):
    print("Checking message")
    self.s.send(password.encode())
    recvdata=self.s.recv(20)
    print(recvdata)
    if(b'Authorized' in recvdata):
        self.serverroot=(recvdata.split(b"$")[1]).decode().strip()
        return True
    else:
        return False
def SetErrorEventHandler(self,event):
    self.errorevent=event
def ErrorEventHandler(self,Message):
    self.errorevent("Error",Message)



def shutDown(self):
    self.s.send(b"Shutdown^")

def deleteFile(self,path):
    self.s.send(b'Delete File $'+(path+"^").encode())
def deleteFolder(self,path):
    self.s.send(b'Delete Folder $'+(path+"^").encode())
    def Rename(self,oldpath,newpath):
        self.s.send(('Rename $'+oldpath+"$"+newpath+"^").encode())
    def GetDirList(self,SPath):
        self.s.send(("Get Dir :"+SPath+"^").encode())
        Err=self.checkErr()
        if(False):
            print("Error: "+Err)
        else:
            dirstr=self.getServerResponse("^")


            dirlist=(dirstr).split(';')
            return dirlist


    def SendFile(self,cpath,spath):
        chunk=1024
        cpath=cpath.replace("\\\\","\\")
        FileName=cpath.split("\\")[-1]
        FileSize=os.path.getsize(cpath.replace("\\\\","\\"))
        print("sending file",FileName)
        self.s.send(("Sending File
$"+spath+"$"+FileName+"$"+str(FileSize)+"^").encode())
        f=open(cpath,'rb')
        x=True
        ByteCount=0
        while FileSize-ByteCount>=chunk:
            data=f.read(chunk)
            self.s.send(data)
            ByteCount+=chunk
    if(FileSize-ByteCount>0):
        self.s.send(f.read(FileSize-ByteCount))
    f.close()

def getServerResponse(self,enclosebyte):
    payload=b''

    byte=self.s.recv(1)
    while byte!=enclosebyte.encode():
        if(byte==enclosebyte.encode()):
            break
        byte=self.s.recv(1)

    byte=self.s.recv(1)

    while byte!=enclosebyte.encode():
        #print(byte.decode(),end='')
        payload+=byte
        byte=self.s.recv(1)

    return payload.decode("UTF-8")



def getFile(self,cpath,fname,spath,size):
    chunks=1024*1024
    recvsize=0
    if(size>0):
        self.s.send(("Get file$"+spath+"^").encode(encoding="UTF-8"))
```

```
            Err=self.checkErr()
            if(Err!="None"):
                self.ErrorEventHandler(Err)
                return
        if(not os.path.exists(cpath.replace(fname,""))):
            os.makedirs(cpath.replace(fname,""))
        print("Writing File",cpath+"\\"+fname)
        with open(cpath+"\\"+fname,"w+") as f:
            pass
        with open(cpath+"\\"+fname,"wb") as f:


            while size-recvsize>0:
                #print(size-bytecount,chunks)
                data=self.s.recv(chunks)
                recvsize+=len(data)
                #print(data)
                f.write(data)
                #self.ProgressEventHandler("Fetching File "+fname,recvsize*100/size)
##              if(size-bytecount>0):
##
##                  print(size-bytecount)
##                  f.write(self.s.recv(size-bytecount))
        print(recvsize,size)
        if(recvsize!=size):
            self.ErrorEventHandler("Some Error Occured")



    def getFolderMetaData(self,spath,cpath):
```

```
    self.s.send(("Get Folder$"+spath+"^").encode())
    Err=self.checkErr()
    if(Err=="None"):
        print("FurtherInfo")
        FileNameStr=self.getServerResponse("^")
        print(FileNameStr)
        Dirs=[]
        Files={}
        sep_string=";*&#%@"

        for i in FileNameStr.split(sep_string):
            #print("Determining")
            if(i!=""):
                data=i.split("$")
                if(len(data)==1):
                    Dirs.append(i.replace("\\\\","\\"))
                else:
                    Files[data[0].replace("\\\\","\\")]=int(data[1])
        return Dirs,Files
    else:
        self.ErrorEventHandler(Err)

def getFolder(self,spath,cpath):
    self.s.send(("Get Folder$"+spath+"^").encode())
    Err=self.checkErr()
    if(Err=="None"):
        print("FurtherInfo")
        FileNameStr=self.getServerResponse("^")
        print(FileNameStr)
        Dirs=[]
        Files={}
        sep_string=";*&#%@"

        for i in FileNameStr.split(sep_string):
            #print("Determining")
            if(i!=""):
                data=i.split("$")
                if(len(data)==1):
                    Dirs.append(i.replace("\\\\","\\"))
                else:
                    Files[data[0].replace("\\\\","\\")]=int(data[1])
        for i in Dirs:
            newcpath=cpath+"\\"+i.replace("\\".join(spath.split("\\")[:-
1]),"").lstrip("\\")
            print(newcpath)
            print(newcpath,os.path.exists(newcpath))
            if(not os.path.exists(newcpath)):
                os.makedirs(newcpath)
                print("Making Dir",i)


        for i in Files:
            #xprint(i.replace("\\".join(spath.split("\\")[:-1]),"").lstrip("\\"))
            newcpath=cpath+"\\"+(i.replace("\\".join(spath.split("\\")[:-
1]),"").lstrip("\\")).replace(i.split("\\")[-1],"")
            print("requesting ",i)
            self.getFile(newcpath,i.split("\\")[-1],i,Files[i])
```

```
    else:
        self.ErrorEventHandler(Err)

def checkErr(self):
    Err=self.getServerResponse("^")
    if Err=="None":
        return "None"
    else:
        return Err

def createDir(self,path):
    self.s.send(b"Create Dir $"+(path+"^").encode())

def createFile(self,path):
    self.s.send(b"Create file $"+(path+"^").encode())

def getLayeredDir(self,path):
    dirs=[]
    try:
        indirs=os.listdir(path)
    except:
        return
    for i in indirs:

            if(os.path.isdir(path+"\\"+i)):
                    templist=self.getLayeredDir(path+"\\"+i)
                    if templist!=None:
                            dirs.extend(templist)
```

```
                dirs.append(path+"\\"+i)
    return dirs

def getFileSize(self,spath):
    self.s.send(("Get File Size$"+spath+"^").encode())
    Err=self.checkErr()
    if(Err=='None'):
        return int(self.getServerResponse("^"))
    else:
        self.ErrorEventHandler(Err)
        return None

def fileLayeredSearch(self,path):
    #print("f "+path)
    try:
        if(path[-1]!="\\"):
            path=path.rstrip()+"\\"
        files=os.listdir(path)
        temp=[]
        for i in files: temp.append(path+"\\"+i)
        files=temp
        #print("asdfad")
    except:
        return
    for i in files:
        if os.path.isdir(i):
            morefiles=self.fileLayeredSearch(i)
            if(morefiles!=None):
                    files.extend(morefiles)
        return files

    def SendFolder(self,cpath,spath):
        print(f"Sending {cpath} to {spath}")
        Files=self.fileLayeredSearch(cpath)
        SendStr="Sending Folder $"+spath+"$"+cpath.split("\\")[-1]+"$"
        FileCode="
        DirCode="
        for i in Files:
            sdata=i.replace("\\".join(cpath.split("\\")[:-1]),spath)
            print(f"sdata={sdata}")
            #print(f"{i}.replace({cpath},{spath})={sdata}")
            if(os.path.isfile(i)):
                FileCode+=sdata.replace("\\\\","\\")+";"+str(os.path.getsize(i))+"|"
            else:
                DirCode+=sdata.replace("\\\\","\\")+"|"
        SendStr+=DirCode
        SendStr+="$"
        SendStr+=FileCode
        self.s.send((SendStr+"^").encode())

        rsps=self.getServerResponse("^")
        while rsps!="$$$$$$$$":
            FileName=rsps
            print("Getting ",FileName)
            c_get_path=FileName.replace(spath,cpath.replace(cpath.split("\\")[-
1],""))
```

```
            f_get_path='\\'.join(FileName.split("\\")[:-1])
            print(f'send ing file {c_get_path} to {f_get_path}')
            self.SendFile(c_get_path,f_get_path)
            rsps=self.getServerResponse("^")


        #print(FileCode,DirCode)


    def main(self):
        while True:
            cmd=input(">>> ")
            if(cmd.startswith("get file")):
                self.getFile(cmd.split('-')[2],(cmd.split('-')[1]).split("\\")[-1],cmd.split('-
')[1],self.getFileSize(cmd.split('-')[1]))
            elif(cmd.startswith("get dir")):
                print(self.GetDirList(cmd.split('-')[1]))
            elif(cmd.startswith("shutdown")):
                self.shutDown()
            elif(cmd.startswith("send file")):
                self.SendFile(cmd.split('-')[1],cmd.split('-')[2])
            elif(cmd.startswith("send folder")):
                self.SendFolder(cmd.split('-')[1],cmd.split('-')[2])
            elif(cmd.startswith("delete file")):
                self.deleteFile(cmd.split('-')[1])
            elif(cmd.startswith("get folder")):
                self.getFolder(cmd.split('-')[1],cmd.split('-')[2])
            else:
                self.s.send(cmd.encode())
```

ClientGUI.py:

```
import tkinter as tk

from tkinter import messagebox

import Client as cl

from tkinter import ttk

import os

from multiprocessing.dummy import Pool,Process

import shutil


client=None

serveraddr=''

clientroot=''

serverport=''

class Login:

    def __init__(self):

        self.root=tk.Tk()

        self.root.title("Login")

        self.root.geometry('510x240+500+500')

        self.root.config(bg="#827b96")

        self.root.resizable(False,False)


        self.IPVar=tk.StringVar()

        self.PortVar=tk.StringVar()

        self.PassVar=tk.StringVar()

        self.ClRootVar=tk.StringVar()
        lbl=tk.Label(self.root,text="Login to Media
Stream",anchor='center',width=60,pady=10,font=('Helvica',10),fg='black',bg="#
827b96")

        lbl.grid(row=1,columnspan=3)

        iplbl=tk.Label(self.root,text="Server
IP:",width=11,fg="black",anchor='w',pady=10,bg="#827b96")

        portlbl=tk.Label(self.root,text="Server
Port:",width=11,anchor='w',pady=10,bg="#827b96")

passlbl=tk.Label(self.root,text="Password:",width=11,anchor='w',pady=10,bg="
#827b96")

        clrootlbl=tk.Label(self.root,text="Client
Root:",width=11,anchor='w',pady=10,bg="#827b96")

        iplbl.grid(row=2)

        portlbl.grid(row=3)

        passlbl.grid(row=4)

        clrootlbl.grid(row=5)


        ip=tk.Entry(self.root,width=60,textvariable=self.IPVar)

        port=tk.Entry(self.root,width=60,textvariable=self.PortVar)

        passwd=tk.Entry(self.root,width=60,textvariable=self.PassVar)

        clroot=tk.Entry(self.root,width=60,textvariable=self.ClRootVar)

        ip.grid(row=2,column=2,columnspan=2)

        port.grid(row=3,column=2,columnspan=2)

        passwd.grid(row=4,column=2,columnspan=2)

        clroot.grid(row=5,column=2,columnspan=2)


        btn=tk.Button(self.root,text="Login",command=self.Log,width=20)

        btn.grid(row=6,columnspan=3,pady=10)
```

```
    def Log(self):
        global serveraddr
        global serverport
        global client
        global clientroot
        try:
            client=cl.Client(self.IPVar.get(),int(self.PortVar.get()))
            if(client.authorize(self.PassVar.get())):
                serveraddr=self.IPVar.get()
                serverport=self.PortVar.get()
                messagebox.showinfo("Success","Successfully Logged in!")
                serveraddr=self.IPVar.get()
                serverport=self.PortVar.get()
                clientroot=self.ClRootVar.get()
                self.root.destroy()
                MainWindow()
            else:
                self.PassVar.set("")
                messagebox.showerror("Error","Wrong Password")
        except:
            messagebox.showerror("Error","Could not connect to
"+self.IPVar.get()+"... Is the IP and port valid?")


class MainWindow:
    def __init__(self):
        self.root=tk.Tk()
        self.root.resizable(False, False)
        self.root.geometry("1150x750")
        self.root.title("MainWindow")
        self.root.config(bg="#827b96")
        self.CurServerPath=client.serverroot
        self.CurClientPath=clientroot
        self.CurServerDirs=[]
        self.CurServerFiles=[]
        self.CurClientDirs=[]
        client.SetErrorEventHandler(self.ErrorEventHandler)
        self.LoadWidgets()



    def ErrorEventHandler(self,Title,Message):
        messagebox.showerror(Title,Message)



    def LoadMainFrames(self):
        lbl=tk.Label(self.root,text="Connected to Server "+serveraddr+" on Port "+serverport,bg="#827b96",font=("Helvica",16))
        lbl.grid(row=1,columnspan=2)



        self.ServerFrame=tk.Frame(self.root,bg='#ababab',height=500,width=450)
        self.ServerFrame.grid(row=2,column=1,padx=5,pady=5)
        self.ClientFrame=tk.Frame(self.root,bg='#ababab',height=500,width=450)
        self.ClientFrame.grid(row=2,column=2,padx=5,pady=5)
        self.ProgressFrame=tk.Frame(self.root,height=10,width=1129)
        self.ProgressFrame.grid(row=3,columnspan=3)

        self.ProgressList=tk.Listbox(self.ProgressFrame,height=5,width=185)
        self.ProgressList.pack(fill='x',side='left')

        self.ProgressScroll=tk.Scrollbar(self.ProgressFrame,orient='vertical')
        self.ProgressScroll.config(command=self.ProgressList.yview)
        self.ProgressScroll.pack(side='right',fill='y')
        self.ProgressList.config(yscrollcommand=self.ProgressScroll.set)

    def AddProgress(self,data):
        self.ProgressList.insert(tk.END,data)

    def LoadSubFrames(self):
        self.CNameLbl=tk.Label(self.ClientFrame,text="Client
Directory",font=('Helvica',10),bg='#5e5e5e',fg='white')
        self.CNameLbl.pack(fill='x')
        self.SNameLbl=tk.Label(self.ServerFrame,text="Server
Directory",font=('Helvica',10),bg='#5e5e5e',fg='white')
        self.SNameLbl.pack(fill='x')

        self.SDirFrame=tk.Frame(self.ServerFrame)
        self.SDirFrame.pack()
        self.CDirFrame=tk.Frame(self.ClientFrame)
        self.CDirFrame.pack()
        self.SDir=tk.StringVar()
        self.CDir=tk.StringVar()

self.ServerDir=tk.Label(self.SDirFrame,textvariable=self.SDir,bg='#a3a3a3',anch
or='w',width=67)#74)

self.ClientDir=tk.Label(self.CDirFrame,textvariable=self.CDir,bg='#a3a3a3',anch
or='w',width=67)
        self.ServerDir.grid(row=1,column=1)
        self.ClientDir.grid(row=1,column=1)


        self.CDir.set(self.CurClientPath)
        self.SDir.set(self.CurServerPath)



self.SBackDir=tk.Button(self.SDirFrame,text="back",command=self.BackDirServer)

self.CBackDir=tk.Button(self.CDirFrame,text="back",command=self.BackDirClient)
        self.SBackDir.grid(row=1,column=3)
        self.CBackDir.grid(row=1,column=3)



self.SRefreshDir=tk.Button(self.SDirFrame,text="Refresh",command=self.LoadServ
erBrowser)

self.CRefreshDir=tk.Button(self.CDirFrame,text="Refresh",command=self.LoadClie
ntBrowser)
        self.SRefreshDir.grid(row=1,column=2)
        self.CRefreshDir.grid(row=1,column=2)


        self.SBrowseFrame=tk.Frame(self.ServerFrame,height=500)
```

```
    self.CBrowseFrame=tk.Frame(self.ClientFrame,height=500)
    self.SBrowseFrame.pack(fill='x')
    self.CBrowseFrame.pack(fill='x')

def LoadBrowseFrames(self):
    self.SBrowse=tk.Listbox(self.SBrowseFrame,width=90,height=30)
    self.SBrowse.pack(side='left')
    self.CBrowse=tk.Listbox(self.CBrowseFrame,width=90,height=30)
    self.CBrowse.pack(side='left')

    self.CBrowseScroll=tk.Scrollbar(self.CBrowseFrame,orient='vertical')
    self.CBrowseScroll.config(command=self.CBrowse.yview)
    self.CBrowseScroll.pack(side='right',fill='y')
    self.SBrowseScroll=tk.Scrollbar(self.SBrowseFrame,orient='vertical')
    self.SBrowseScroll.config(command=self.SBrowse.yview)
    self.SBrowseScroll.pack(side='right',fill='y')

    self.SBrowse.config(yscrollcommand=self.SBrowseScroll.set)
    self.CBrowse.config(yscrollcommand=self.CBrowseScroll.set)

    self.SBrowse.bind('<<ListboxSelect>>',self.SBrowseSelectionChanged)
    self.CBrowse.bind('<<ListboxSelect>>',self.CBrowseSelectionChanged)
    self.SBrowse.bind('<Double-Button>',self.SBrowseDClick)
    self.CBrowse.bind('<Double-Button>',self.CBrowseDClick)

    self.CControl=tk.Frame(self.ClientFrame,bg='#bdbdbd',height=60)
    self.SControl=tk.Frame(self.ServerFrame,bg='#bdbdbd',height=60)
    self.CControl.pack(side='bottom',fill='x',pady=5)
```

```
        self.SControl.pack(side='bottom',fill='x',pady=5)

    def LoadControlFrames(self):
        self.SFolderFrame=tk.LabelFrame(self.SControl,text="Folder Options")
        self.CFolderFrame=tk.LabelFrame(self.CControl,text="Folder Options")
        self.SFolderFrame.pack(side='left',padx=5,pady=5)
        self.CFolderFrame.pack(side='left',padx=5,pady=5)

        self.SFileFrame=tk.LabelFrame(self.SControl,text="File Options")
        self.CFileFrame=tk.LabelFrame(self.CControl,text="File Options")
        self.SFileFrame.pack(side='right',padx=5,pady=5)
        self.CFileFrame.pack(side='right',padx=5,pady=5)

        self.S2CFile=tk.Button(self.SFileFrame,text="Get File From
Server",command=self.S2CFileEvent)
        self.C2SFile=tk.Button(self.CFileFrame,text="Send File To
Server",command=self.C2SFileEvent)
        self.S2CFile.grid(row=1,column=1,padx=5,columnspan=2)
        self.C2SFile.grid(row=1,column=1,padx=5,columnspan=2)

        self.S2CFolder=tk.Button(self.SFolderFrame,text="Get Folder From
Server",command=self.S2CFolderEvent)
        self.C2SFolder=tk.Button(self.CFolderFrame,text="Send Folder To
Server",command=self.C2SFolderEvent)
        self.S2CFolder.grid(row=1,column=1,padx=5,columnspan=2)
        self.C2SFolder.grid(row=1,column=1,padx=5,columnspan=2)

        self.SFileDelete=tk.Button(self.SFileFrame,text="Delete
File",command=self.SFileDeleteEvent)
```

```
        self.CFileDelete=tk.Button(self.CFileFrame,text="Delete
File",command=self.CFileDeleteEvent)
        self.SFileDelete.grid(row=1,column=3,padx=5)
        self.CFileDelete.grid(row=1,column=3,padx=5)

        self.SFolderDelete=tk.Button(self.SFolderFrame,text="Delete
Folder",command=self.SFolderDeleteEvent)
        self.CFolderDelete=tk.Button(self.CFolderFrame,text="Delete
Folder",command=self.CFolderDeleteEvent)
        self.SFolderDelete.grid(row=1,column=3,padx=5)
        self.CFolderDelete.grid(row=1,column=3,padx=5)

        self.SFileRenameLbl=tk.Label(self.SFileFrame,text="Rename:")
        self.SFileRenameLbl.grid(row=2,column=1,pady=5)
        self.CFileRenameLbl=tk.Label(self.CFileFrame,text="Rename:")
        self.CFileRenameLbl.grid(row=2,column=1,pady=5)

        self.SFolderRenameLbl=tk.Label(self.SFolderFrame,text="Rename:")
        self.SFolderRenameLbl.grid(row=2,column=1,pady=5)
        self.CFolderRenameLbl=tk.Label(self.CFolderFrame,text="Rename:")
        self.CFolderRenameLbl.grid(row=2,column=1,pady=5)

        self.SFileRename=tk.StringVar()
        self.CFileRename=tk.StringVar()
        self.SFolderRename=tk.StringVar()
        self.CFolderRename=tk.StringVar()


self.SFileRenameEntry=tk.Entry(self.SFileFrame,textvariable=self.SFileRename)
```

```
        self.SFileRenameEntry.grid(row=2,column=2)

self.CFileRenameEntry=tk.Entry(self.CFileFrame,textvariable=self.CFileRename)
        self.CFileRenameEntry.grid(row=2,column=2)



self.SFolderRenameEntry=tk.Entry(self.SFolderFrame,textvariable=self.SFolderRen
ame)
        self.SFolderRenameEntry.grid(row=2,column=2)

self.CFolderRenameEntry=tk.Entry(self.CFolderFrame,textvariable=self.CFolderRe
name)
        self.CFolderRenameEntry.grid(row=2,column=2)



self.SFileRenameBtn=tk.Button(self.SFileFrame,text="Apply",command=self.SFileR
enameEvent)
        self.SFileRenameBtn.grid(row=2,column=3)

self.CFileRenameBtn=tk.Button(self.CFileFrame,text="Apply",command=self.CFileR
enameEvent)
        self.CFileRenameBtn.grid(row=2,column=3)



self.SFolderRenameBtn=tk.Button(self.SFolderFrame,text="Apply",command=self.
SFolderRenameEvent)
        self.SFolderRenameBtn.grid(row=2,column=3)

self.CFolderRenameBtn=tk.Button(self.CFolderFrame,text="Apply",command=self.
CFolderRenameEvent)
        self.CFolderRenameBtn.grid(row=2,column=3)
```

```
def LoadWidgets(self):
    self.LoadMainFrames()
    self.LoadSubFrames()
    self.LoadBrowseFrames()
    self.LoadControlFrames()

    self.LoadClientBrowser()
    self.LoadServerBrowser()
    self.SFolderFrameLockDown()
    self.SFileFrameLockDown()
    self.CFolderFrameLockDown()
    self.CFileFrameLockDown()

def SFileRenameEvent(self):
    oldName=self.FullCurSelectedPathServer()
    ext=oldName.split(".")[-1]
    newName=self.CurServerPath+"\\"+self.SFileRename.get()+'.'+ext
    client.Rename(oldName,newName)
    self.SFileRename.set("")
    self.LoadServerBrowser()
    self.AddProgress(f"rename {oldName} to {newName} on Server")
    messagebox.showinfo("Success","The File name was changed successfully")

def CFileRenameEvent(self):
    oldName=self.FullCurSelectedPathClient()
    ext=oldName.split(".")[-1]
    newName=self.CurClientPath+"\\"+self.CFileRename.get()+"."+ext
```

```
        os.rename(oldName,newName)
        self.CFileRename.set("")
        self.LoadClientBrowser()
        self.AddProgress(f"rename {oldName} to {newName} On Client")
        messagebox.showinfo("Success","The File name was changed successfully")


    def SFolderRenameEvent(self):
        oldName=self.FullCurSelectedPathServer()
        newName=self.CurServerPath+"\\"+self.SFolderRename.get()
        client.Rename(oldName,newName)
        self.SFolderRename.set("")
        self.LoadServerBrowser()
        self.AddProgress(f"rename {oldName} to {newName} on Server")
        messagebox.showinfo("Success","The Directory name was changed
successfully")

    def CFolderRenameEvent(self):
        oldName=self.FullCurSelectedPathClient()
        newName=self.CurClientPath+"\\"+self.CFolderRename.get()
        os.rename(oldName,newName)
        self.LoadClientBrowser()
        self.AddProgress(f"rename {oldName} to {newName} on Client")
        messagebox.showinfo("Success","The Directory name was changed
successfully")
```

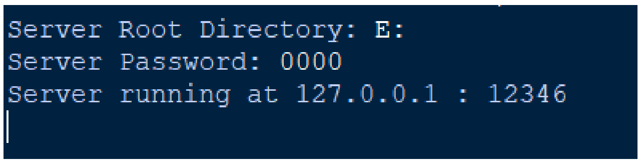
```
    def S2CFileEvent(self):
        #self.ProgressData.set("Fetching file
"+self.SBrowse.get(self.SBrowse.curselection()))

client.getFile(self.CurClientPath,self.SBrowse.get(self.SBrowse.curselection()),self.Ful
lCurSelectedPathServer(),client.getFileSize(self.FullCurSelectedPathServer()))
        self.AddProgress(f"Fetch {self.SBrowse.get(self.SBrowse.curselection())}")
        self.LoadClientBrowser()

    def C2SFileEvent(self):
        client.SendFile(self.FullCurSelectedPathClient(),self.CurServerPath)
        self.AddProgress(f"Send {self.FullCurSelectedPathClient()}")
        self.LoadServerBrowser()



    def S2CFolderEvent(self):
        client.getFolder(self.FullCurSelectedPathServer(),self.CurClientPath)
        self.AddProgress(f"Fetch {self.FullCurSelectedPathServer()}")
        self.LoadClientBrowser()
    def C2SFolderEvent(self):
        #print(f"SendFolder({self.FullCurSelectedPathClient()},{self.CurServerPath})")
        client.SendFolder(self.FullCurSelectedPathClient(),self.CurServerPath)
        self.AddProgress(f"Send {self.FullCurSelectedPathClient()}")
        self.LoadServerBrowser()

    def SFileDeleteEvent(self):
        client.deleteFile(self.FullCurSelectedPathServer())
        self.AddProgress(f"Delete {self.FullCurSelectedPathServer()} on Server" )
        self.LoadServerBrowser()

    def CFileDeleteEvent(self):
        os.remove(self.FullCurSelectedPathClient())
        self.AddProgress(f"Delete {self.FullCurSelectedPathClient()} on Client" )
        self.LoadClientBrowser()

    def SFolderDeleteEvent(self):
        client.deleteFolder(self.FullCurSelectedPathServer())
        self.AddProgress(f"Delete {self.FullCurSelectedPathServer()} on Server" )
        self.LoadServerBrowser()

    def CFolderDeleteEvent(self):
        shutil.rmtree(self.FullCurSelectedPathClient())
        self.AddProgress(f"Delete {self.FullCurSelectedPathClient()} on Client" )
        self.LoadClientBrowser()

    def SetServerPath(self,path):
        self.SDir.set(path)
        self.CurServerPath=path
        self.LoadServerBrowser()

    def SetClientPath(self,path):
        self.CDir.set(path)
        self.CurClientPath=path
        self.LoadClientBrowser()

    def SBrowseDClick(self,evt):
        lb=evt.widget
        if(lb.get(lb.curselection()) in self.CurServerDirs):
            self.SetServerPath(self.FullCurSelectedPathServer())
    def CBrowseDClick(self,evt):
        lb=evt.widget
        if(os.path.isdir(self.FullCurSelectedPathClient())):
            self.SetClientPath(self.FullCurSelectedPathClient())



    def FullCurSelectedPathServer(self):
        if(self.CurServerPath[:-1]=='\\'):
            return self.CurServerPath+self.SBrowse.get(self.SBrowse.curselection())
        return self.CurServerPath+"\\"+self.SBrowse.get(self.SBrowse.curselection())
    def FullCurSelectedPathClient(self):
        if(self.CurClientPath[:-1]=='\\'):
            return self.CurClientPath+self.CBrowse.get(self.CBrowse.curselection())
        return self.CurClientPath+"\\"+self.CBrowse.get(self.CBrowse.curselection())

    def SBrowseSelectionChanged(self,evt):
        lb=evt.widget
        try:
            data=lb.get(lb.curselection())
            if(data in self.CurServerDirs):
                self.SFileFrameLockDown()
                self.SFolderFrameRevive()
            else:
                self.SFolderFrameLockDown()
                self.SFileFrameRevive()
        except:
            pass
    def CBrowseSelectionChanged(self,evt):
        lb=evt.widget
        try:
            data=lb.get(lb.curselection())
            if(os.path.isfile(self.CurClientPath+"\\"+data)):
                self.CFileFrameRevive()
                self.CFolderFrameLockDown()
            else:
                self.CFileFrameLockDown()
                self.CFolderFrameRevive()
        except:
            pass

    def SFileFrameLockDown(self):
        self.S2CFile.config(state='disabled')
        self.SFileDelete.config(state='disabled')
        self.SFileRenameBtn.config(state='disabled')
        self.SFileRenameEntry.config(state='disabled')
        self.SFileRenameLbl.config(state='disabled')
    def SFolderFrameLockDown(self):
        self.S2CFolder.config(state='disabled')
        self.SFolderDelete.config(state='disabled')
        self.SFolderRenameBtn.config(state='disabled')
        self.SFolderRenameEntry.config(state='disabled')
        self.SFolderRenameLbl.config(state='disabled')
    def SFileFrameRevive(self):
        self.S2CFile.config(state='normal')
        self.SFileDelete.config(state='normal')
        self.SFileRenameBtn.config(state='normal')
        self.SFileRenameEntry.config(state='normal')
        self.SFileRenameLbl.config(state='normal')
    def SFolderFrameRevive(self):
        self.S2CFolder.config(state='normal')
        self.SFolderDelete.config(state='normal')
        self.SFolderRenameBtn.config(state='normal')
        self.SFolderRenameEntry.config(state='normal')
        self.SFolderRenameLbl.config(state='normal')
    def CFileFrameLockDown(self):
        self.C2SFile.config(state='disabled')
        self.CFileDelete.config(state='disabled')
        self.CFileRenameBtn.config(state='disabled')
        self.CFileRenameEntry.config(state='disabled')
        self.CFileRenameLbl.config(state='disabled')
    def CFolderFrameLockDown(self):
        self.C2SFolder.config(state='disabled')
        self.CFolderDelete.config(state='disabled')
        self.CFolderRenameBtn.config(state='disabled')
        self.CFolderRenameEntry.config(state='disabled')
        self.CFolderRenameLbl.config(state='disabled')
    def CFileFrameRevive(self):
        self.C2SFile.config(state='normal')
        self.CFileDelete.config(state='normal')
        self.CFileRenameBtn.config(state='normal')
        self.CFileRenameEntry.config(state='normal')
        self.CFileRenameLbl.config(state='normal')
    def CFolderFrameRevive(self):
        self.C2SFolder.config(state='normal')
        self.CFolderDelete.config(state='normal')
        self.CFolderRenameBtn.config(state='normal')
        self.CFolderRenameEntry.config(state='normal')
        self.CFolderRenameLbl.config(state='normal')




    def BackDirServer(self):
        CurDir=self.CurServerPath
        if(len(CurDir.split("\\"))>1):
            CurDir='\\'.join(CurDir.split("\\")[:-1])
            self.SetServerPath(CurDir)
    def BackDirClient(self):
        CurDir=self.CurClientPath
        if(len(CurDir.split("\\"))>1):
            CurDir='\\'.join(CurDir.split("\\")[:-1])
            self.SetClientPath(CurDir)

    def LoadServerBrowser(self):
        self.SFolderFrameLockDown()
        self.SFileFrameLockDown()
        self.SBrowse.delete(0,tk.END)
        self.CirServerFiles=[]
        self.CirServerDirs=[]
        serverdata=client.GetDirList(self.CurServerPath)
```

```
        for i in serverdata:
            if(i!=''):
                if(i[-1]=="f"):
                    self.CurServerFiles.append(i[:-1])
                else:
                    self.CurServerDirs.append(i[:-1])

        for i in serverdata:
            if(i!=''):
                self.SBrowse.insert(tk.END,i[:-1])

    def LoadClientBrowser(self):
        self.CFolderFrameLockDown()
        self.CFileFrameLockDown()

        self.CBrowse.delete(0,tk.END)
        self.CurClientDirs=[x for x in os.listdir(self.CurClientPath) if x!='']
        for i in self.CurClientDirs:
            if(i!=''):
                self.CBrowse.insert(tk.END,i)

Login()
```

SETTING UP SERVER

The server can be setup in the server machine by the following the following steps.

1. Open Server.py on IDLE
2. Run > Run Module
3. Now it will ask for the root directory and password for server. Type it and hit enter.



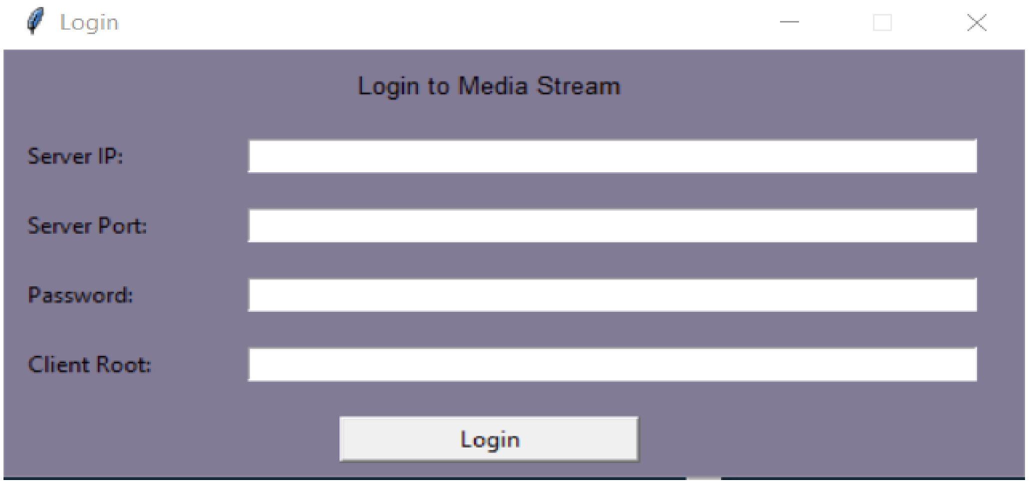
```
Server Root Directory: E:
Server Password: 0000
Server running at 127.0.0.1 : 12346
```

If no errors are shown, the server is up and running.

CONNECTING CLIENT TO SERVER

Client can be connected to the server by the following steps

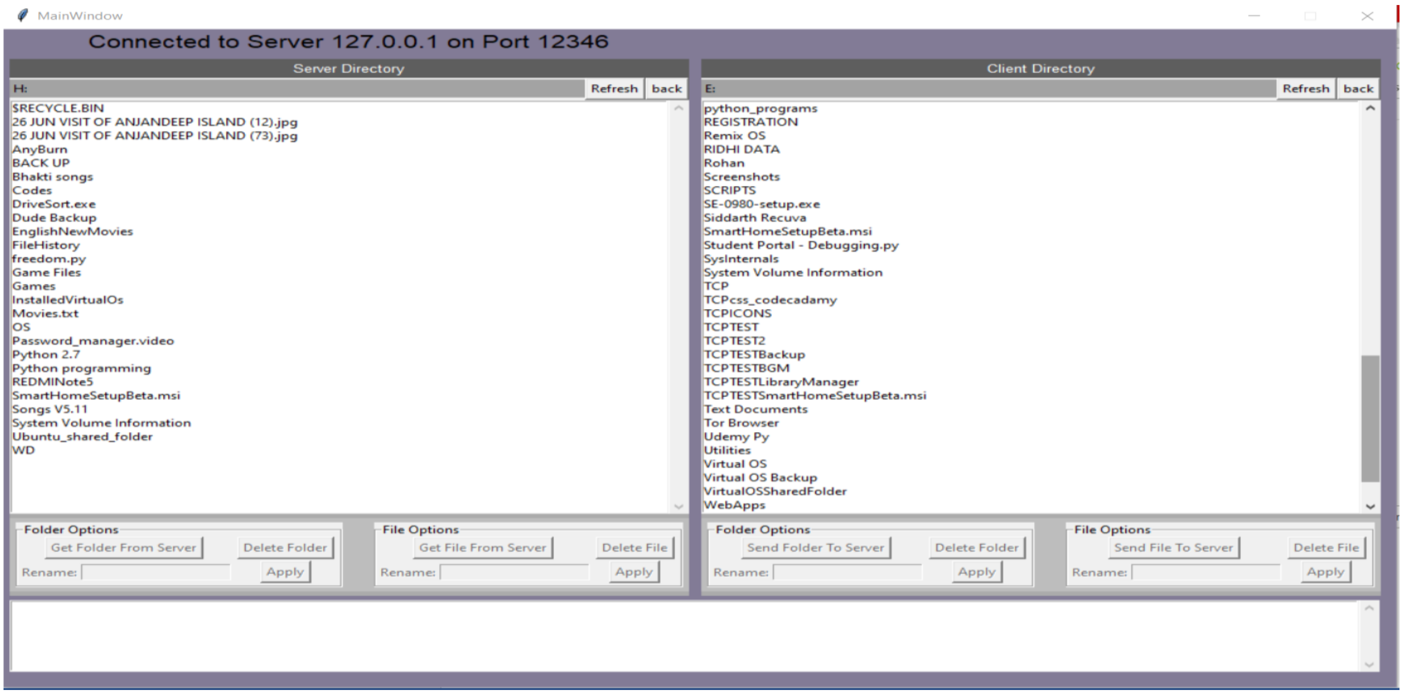
1. Open ClientGUI.py
2. Run > Run Module
3. This gives a prompt to type in server ip, port, password and root working directory of the client. Go ahead and type it



If the ip, port, password are valid, the program starts.

CODE WALKTHROUGH

After proper setting up of the server and client, the client program shows a window like this



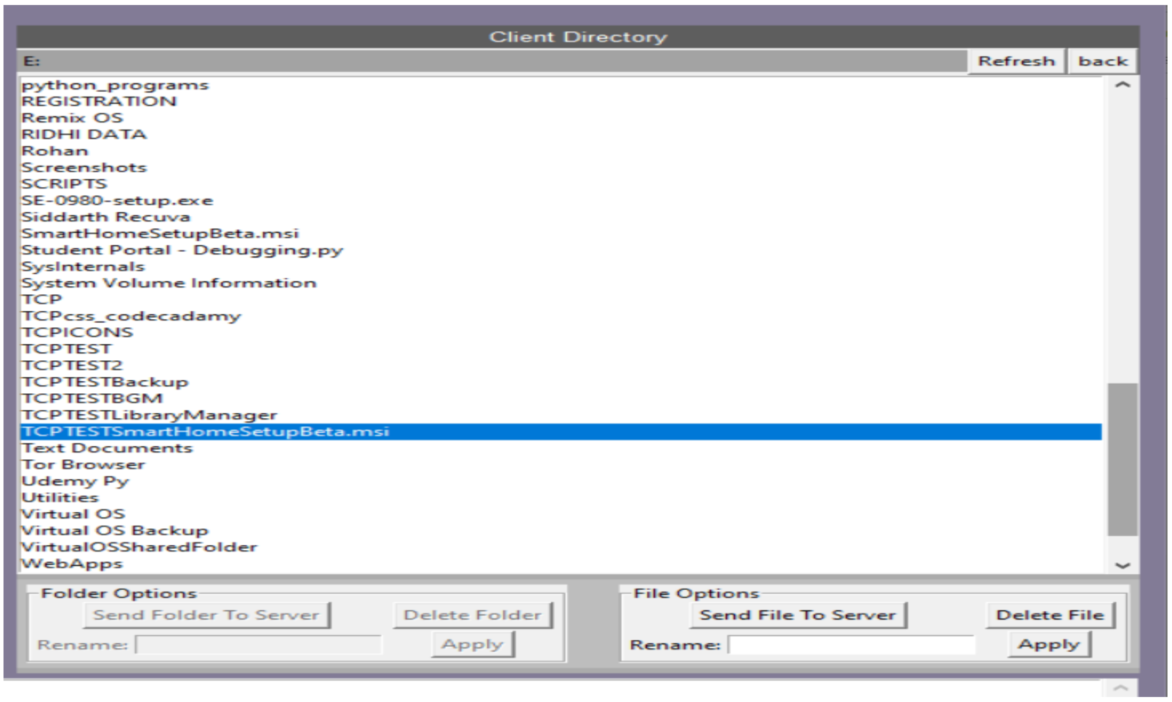
Connected to Server 127.0.0.1 on Port 12346

```
H:
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
Games
InstalledVirtualOs
Movies.txt
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
Ubuntu_shared_folder
WD
```

```
E:
python_programs
REGISTRATION
Remix OS
RIDHI DATA
Rohan
Screenshots
SCRIPTS
SE-0980-setup.exe
Siddarth Recuva
SmartHomeSetupBeta.msi
Student Portal - Debugging.py
SysInternals
System Volume Information
TCP
TCPcss_codecadamy
TCPICONS
TCPTEST
TCPTEST2
TCPTESTBackup
TCPTESTBGM
TCPTESTLibraryManager
TCPTESTSmartHomeSetupBeta.msi
Text Documents
Tor Browser
Udemy Py
Utilities
Virtual OS
Virtual OS Backup
VirtualOSSharedFolder
WebApps
```

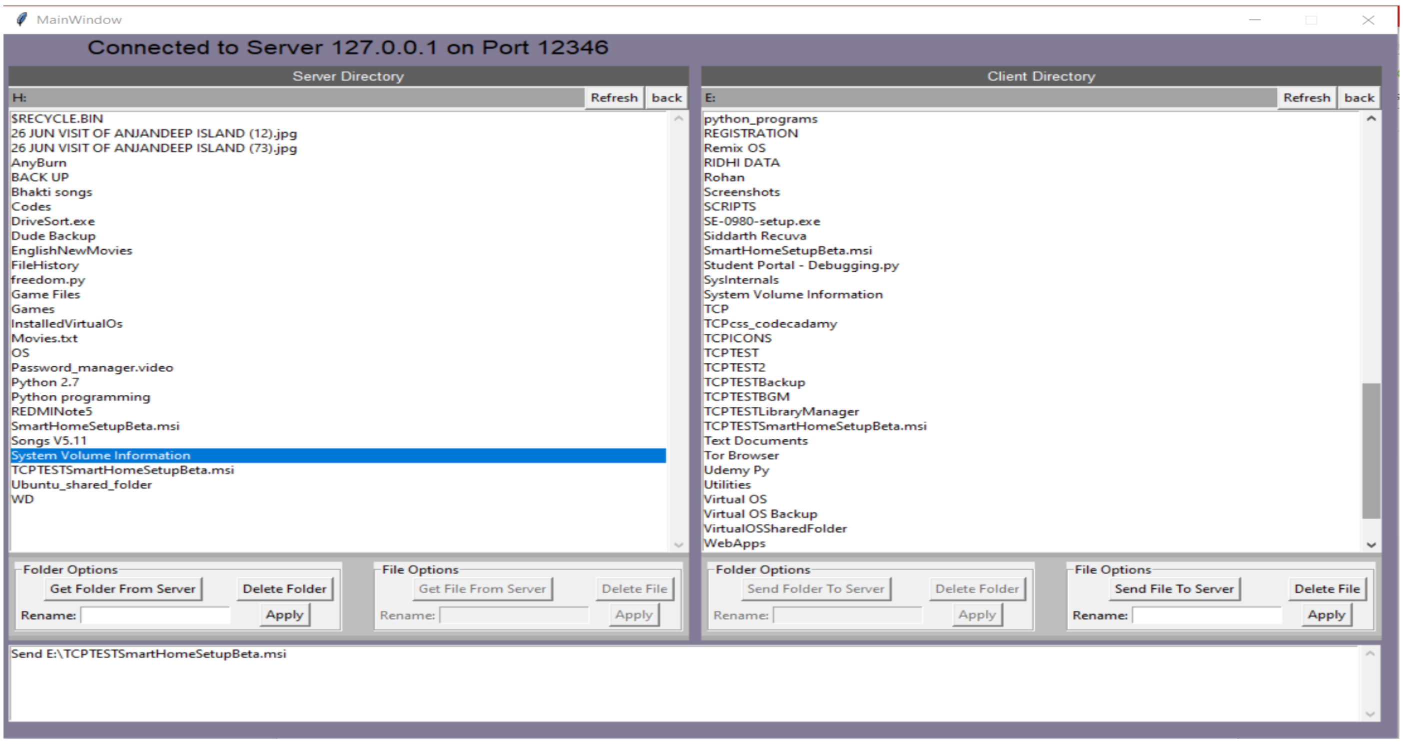
To send a file/folder to a server:

Click on the file on the server directory and click send file to server.



```
E:
python_programs
REGISTRATION
Remix OS
RIDHI DATA
Rohan
Screenshots
SCRIPTS
SE-0980-setup.exe
Siddarth Recuva
SmartHomeSetupBeta.msi
Student Portal - Debugging.py
SysInternals
System Volume Information
TCP
TCPcss_codecadamy
TCPICONS
TCPTEST
TCPTEST2
TCPTESTBackup
TCPTESTBGM
TCPTESTLibraryManager
TCPTESTSmartHomeSetupBeta.msi
Text Documents
Tor Browser
Udemy Py
Utilities
Virtual OS
Virtual OS Backup
VirtualOSSharedFolder
WebApps
```

After this, you'll find the file in the server

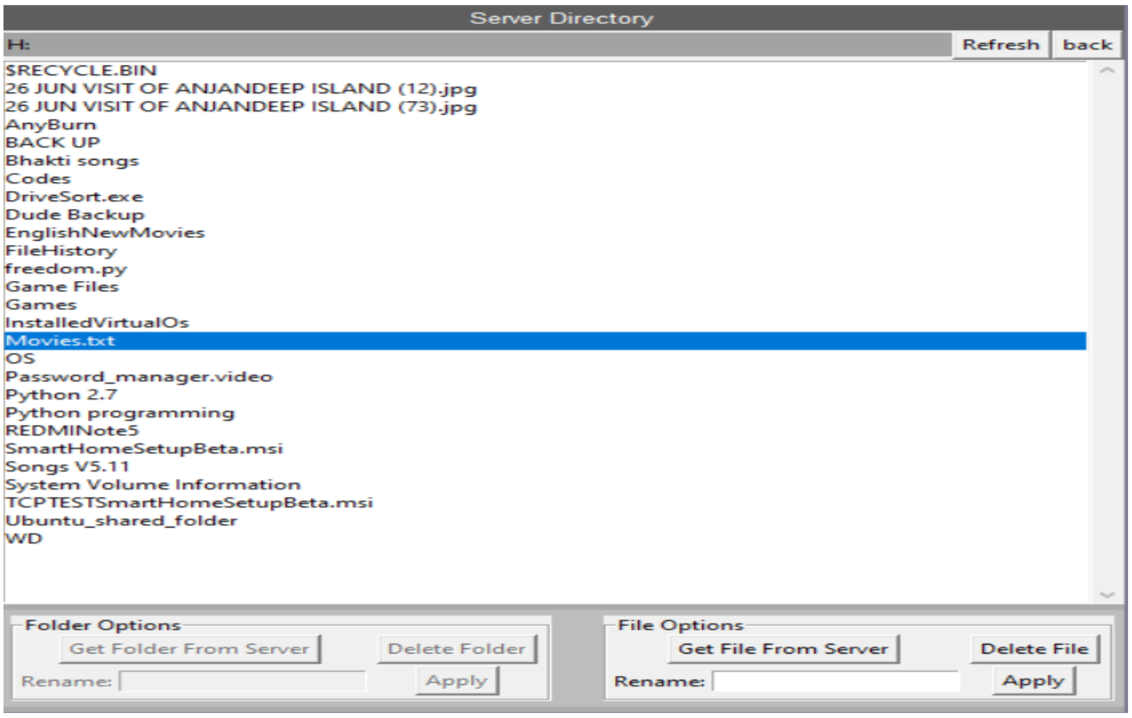


```
Connected to Server 127.0.0.1 on Port 12346

Server Directory
H:
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
Games
InstalledVirtualOs
Movies.txt
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD

Client Directory
E:
python_programs
REGISTRATION
Remix OS
RIDHI DATA
Rohan
Screenshots
SCRIPTS
SE-0980-setup.exe
Siddarth Recuva
SmartHomeSetupBeta.msi
Student Portal - Debugging.py
SysInternals
System Volume Information
TCP
TCPcss_codecadamy
TCPICONS
TCPTEST
TCPTEST2
TCPTESTBackup
TCPTESTBGM
TCPTESTLibraryManager
TCPTESTSmartHomeSetupBeta.msi
Text Documents
Tor Browser
Udemy Py
Utilities
Virtual OS
Virtual OS Backup
VirtualOSSharedFolder
WebApps

Send E:\TCPTESTSmartHomeSetupBeta.msi
```

Same goes for sending folders.

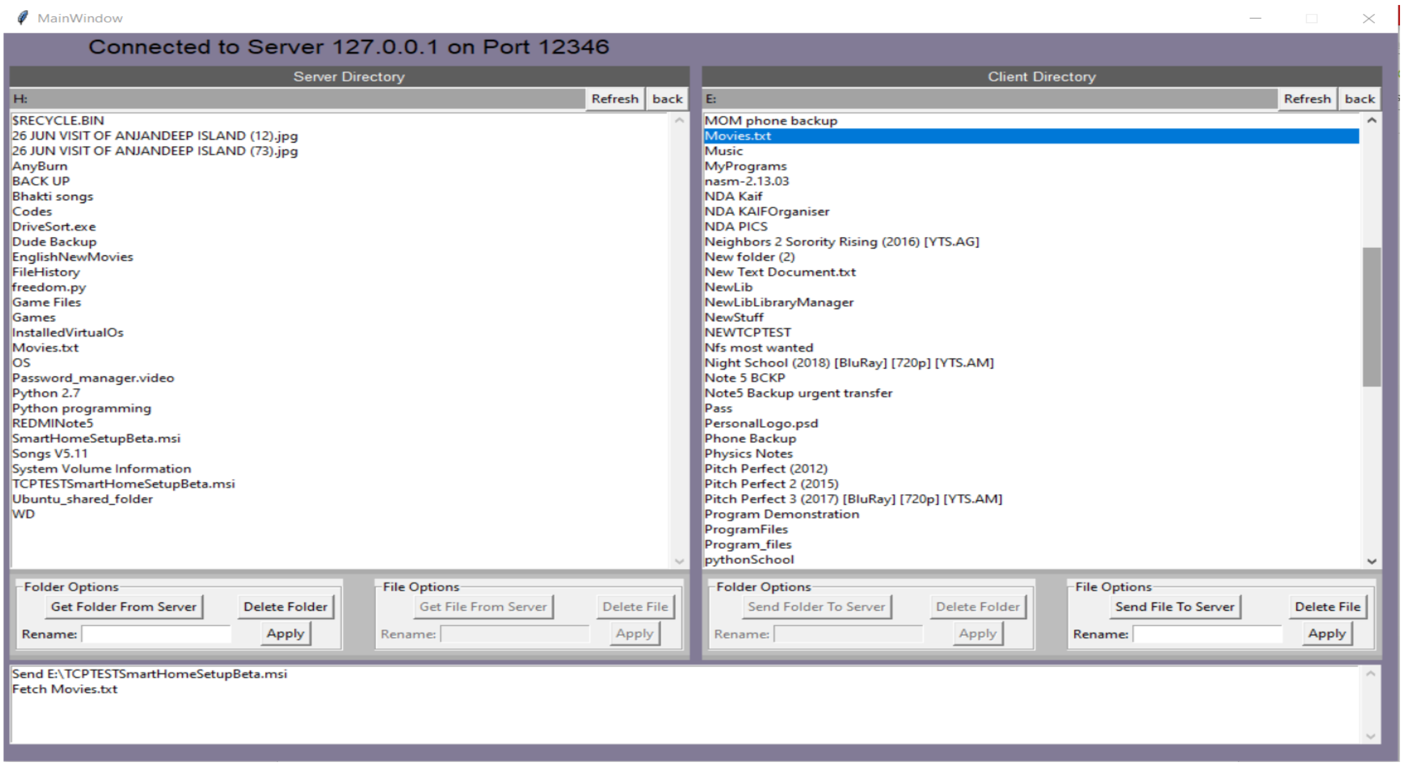
To receive files/folders from server:

Click on the file/folder on the server directory list and click on get file/folder from server.



```
H:
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
Games
InstalledVirtualOs
Movies.txt
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

After this you'll find the file on the client directory list.



Connected to Server 127.0.0.1 on Port 12346

H:

```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
Games
InstalledVirtualOs
Movies.txt
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

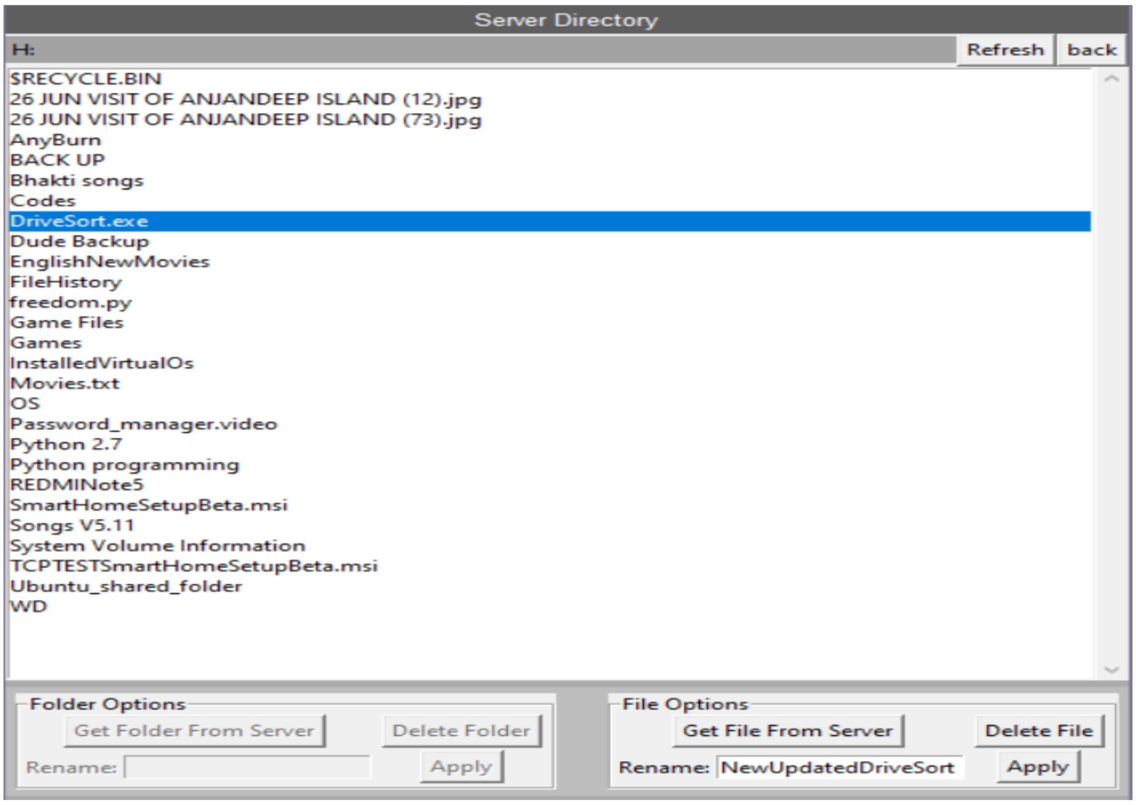
E:

```
MOM phone backup
Movies.txt
Music
MyPrograms
nasm-2.13.03
NDA Kaif
NDA KAIFOrganiser
NDA PICS
Neighbors 2 Sorority Rising (2016) [YTS.AG]
New folder (2)
New Text Document.txt
NewLib
NewLibLibraryManager
NewStuff
NEWTCPTEST
Nfs most wanted
Night School (2018) [BluRay] [720p] [YTS.AM]
Note 5 BCKP
Note5 Backup urgent transfer
Pass
PersonalLogo.psd
Phone Backup
Physics Notes
Pitch Perfect (2012)
Pitch Perfect 2 (2015)
Pitch Perfect 3 (2017) [BluRay] [720p] [YTS.AM]
Program Demonstration
ProgramFiles
Program_files
pythonSchool
```

```
Send E:\TCPTESTSmartHomeSetupBeta.msi
Fetch Movies.txt
```

To Re name a file:

Select the file to be re-named and type in the new name.

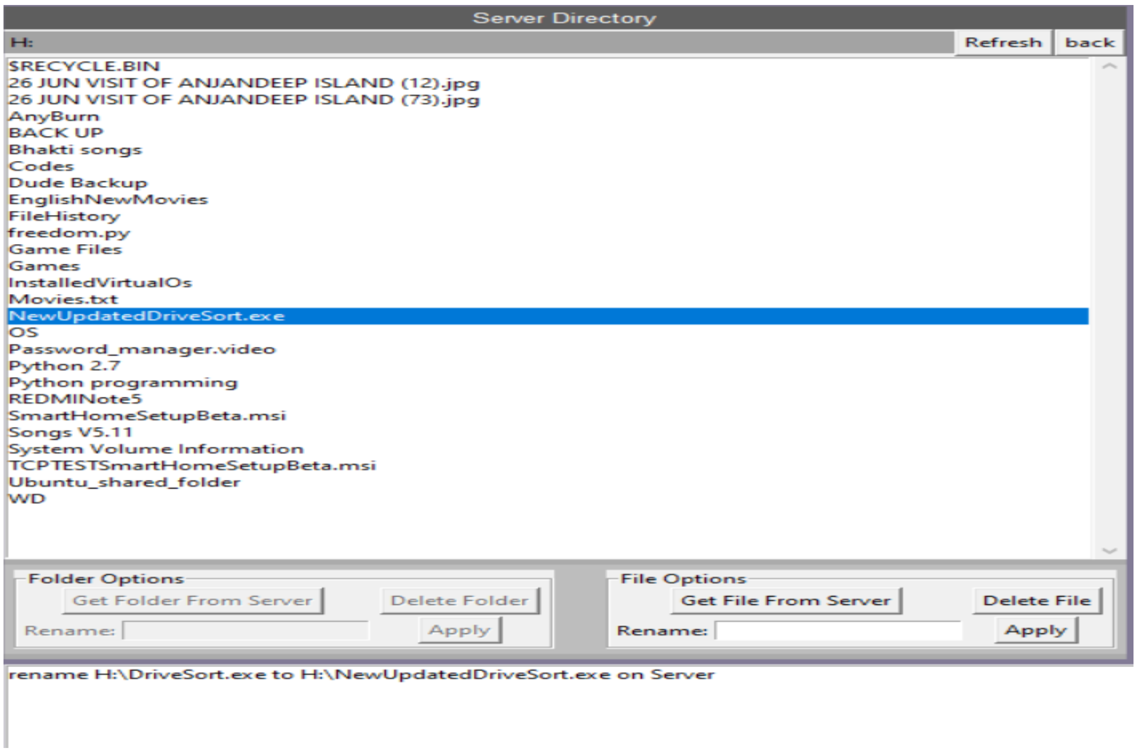


H:

```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
Games
InstalledVirtualOs
Movies.txt
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

Rename: NewUpdatedDriveSort

Now click on the Apply button. You will see the changes reflected on the directory list (shown below).

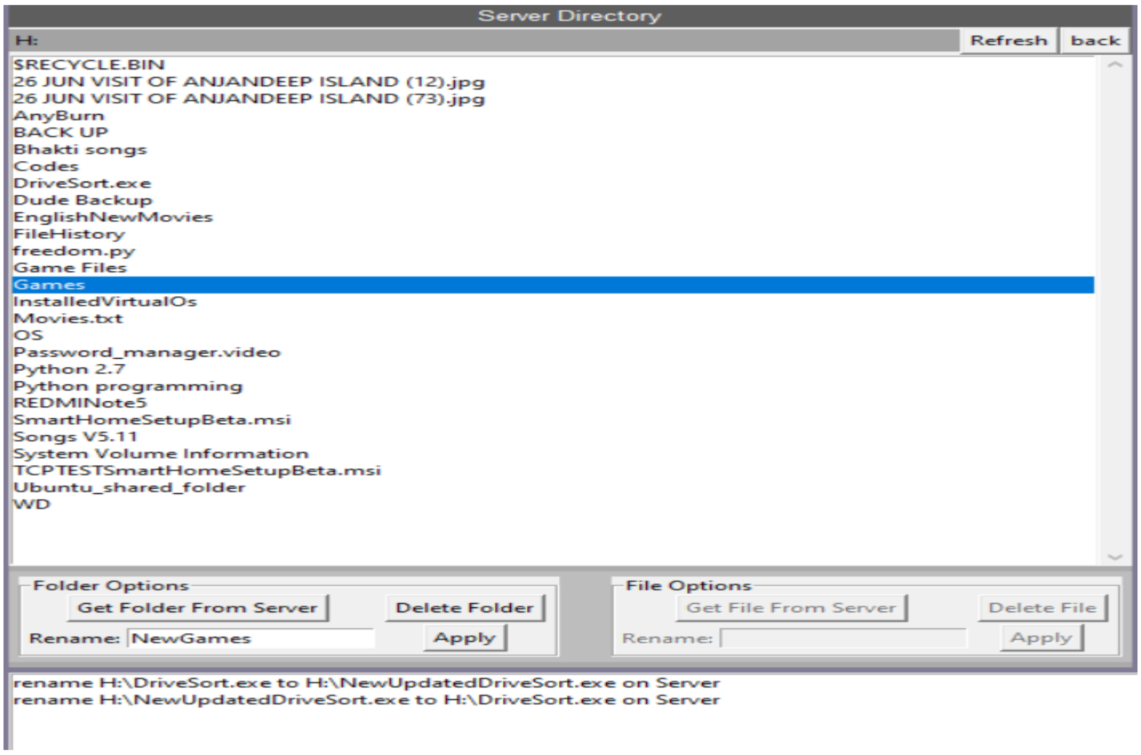


```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
Games
InstalledVirtualOs
Movies.txt
NewUpdatedDriveSort.exe
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

```
rename H:\DriveSort.exe to H:\NewUpdatedDriveSort.exe on Server
```

TO RENAME A FOLDER:

Select the folder to be re-named and then type in the new name of the folder.

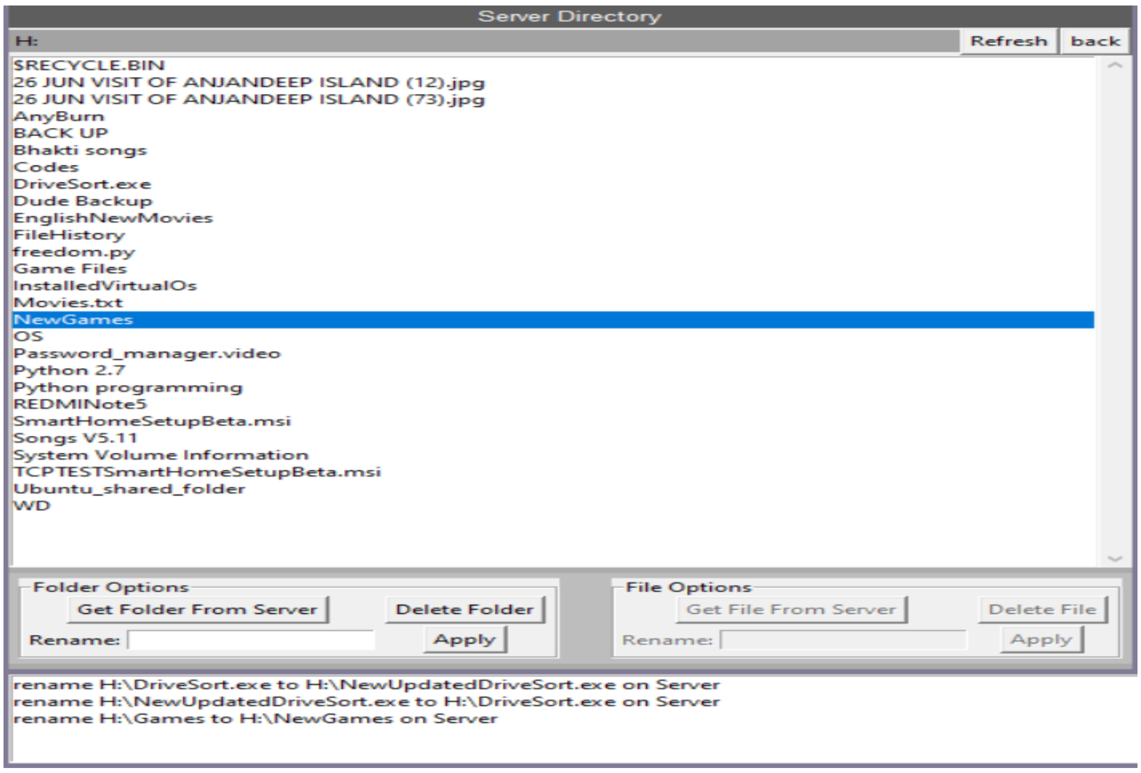


```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
Games
InstalledVirtualOs
Movies.txt
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

Rename: NewGames

```
rename H:\DriveSort.exe to H:\NewUpdatedDriveSort.exe on Server
rename H:\NewUpdatedDriveSort.exe to H:\DriveSort.exe on Server
```

Now click on apply. The change will be reflected shortly. (shown below)

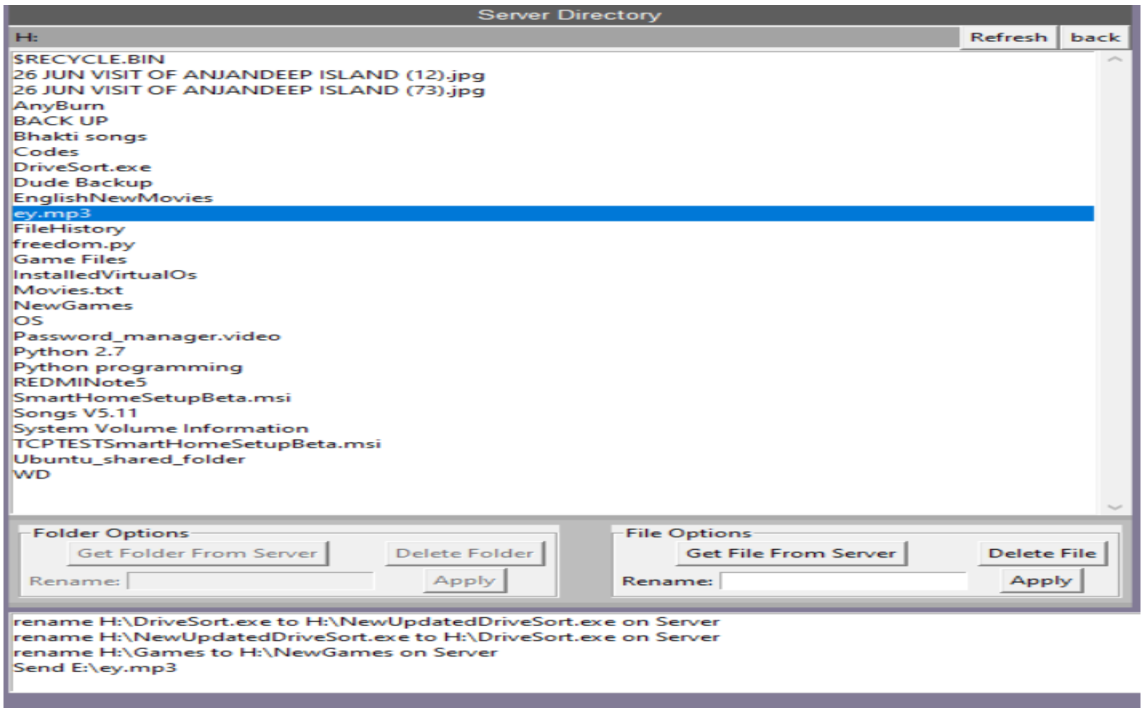


```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
InstalledVirtualOs
Movies.txt
NewGames
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

```
rename H:\DriveSort.exe to H:\NewUpdatedDriveSort.exe on Server
rename H:\NewUpdatedDriveSort.exe to H:\DriveSort.exe on Server
rename H:\Games to H:\NewGames on Server
```

TO DELETE A FILE FROM SERVER:

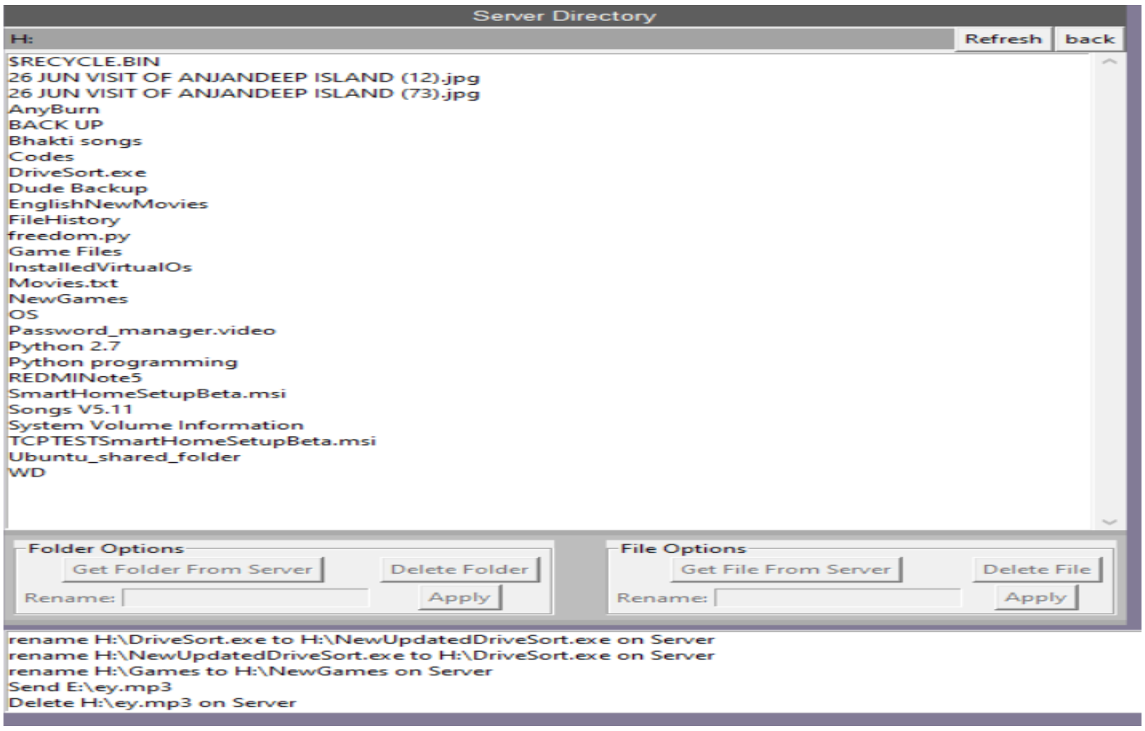
Select the file to be deleted.



```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
ey.mp3
FileHistory
freedom.py
Game Files
InstalledVirtualOs
Movies.txt
NewGames
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

```
rename H:\DriveSort.exe to H:\NewUpdatedDriveSort.exe on Server
rename H:\NewUpdatedDriveSort.exe to H:\DriveSort.exe on Server
rename H:\Games to H:\NewGames on Server
Send E:\ey.mp3
```

Now click on delete file and the file will be deleted from the file and the changes will be reflected shortly (shown below).

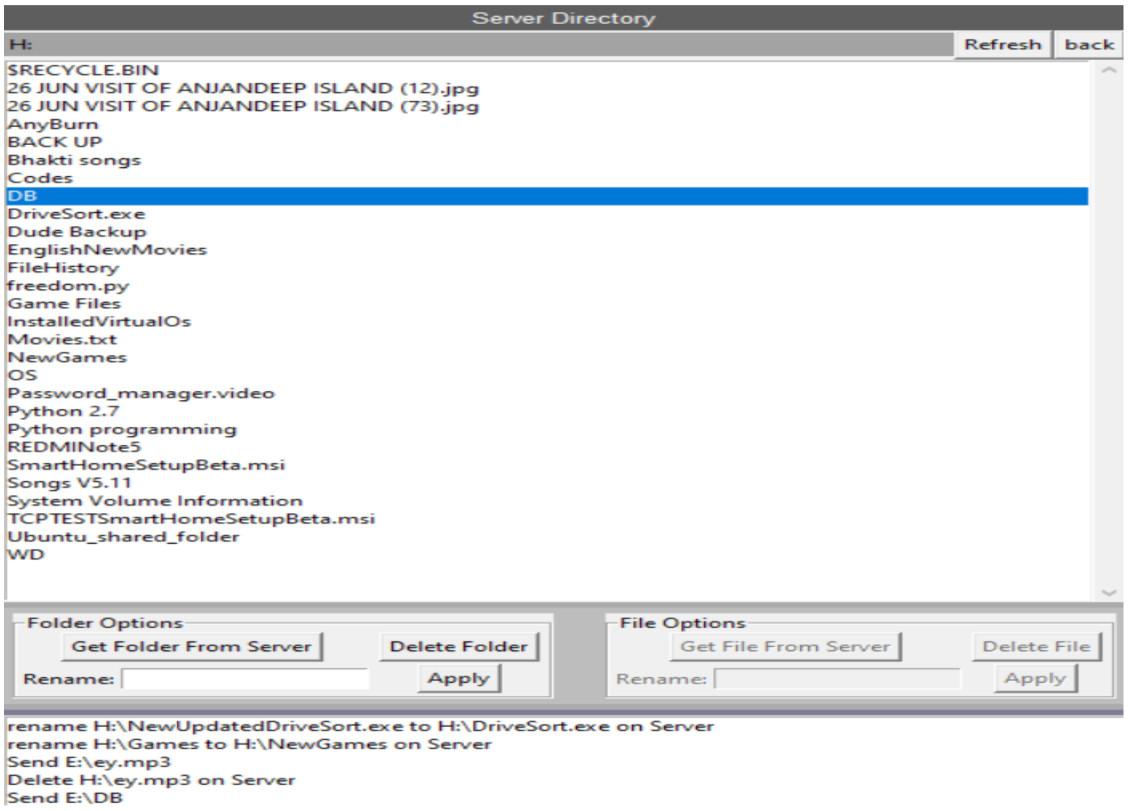


```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
InstalledVirtualOs
Movies.txt
NewGames
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

```
rename H:\DriveSort.exe to H:\NewUpdatedDriveSort.exe on Server
rename H:\NewUpdatedDriveSort.exe to H:\DriveSort.exe on Server
rename H:\Games to H:\NewGames on Server
Send E:\ey.mp3
Delete H:\ey.mp3 on Server
```

TO DELETE FOLDER FROM SERVER:

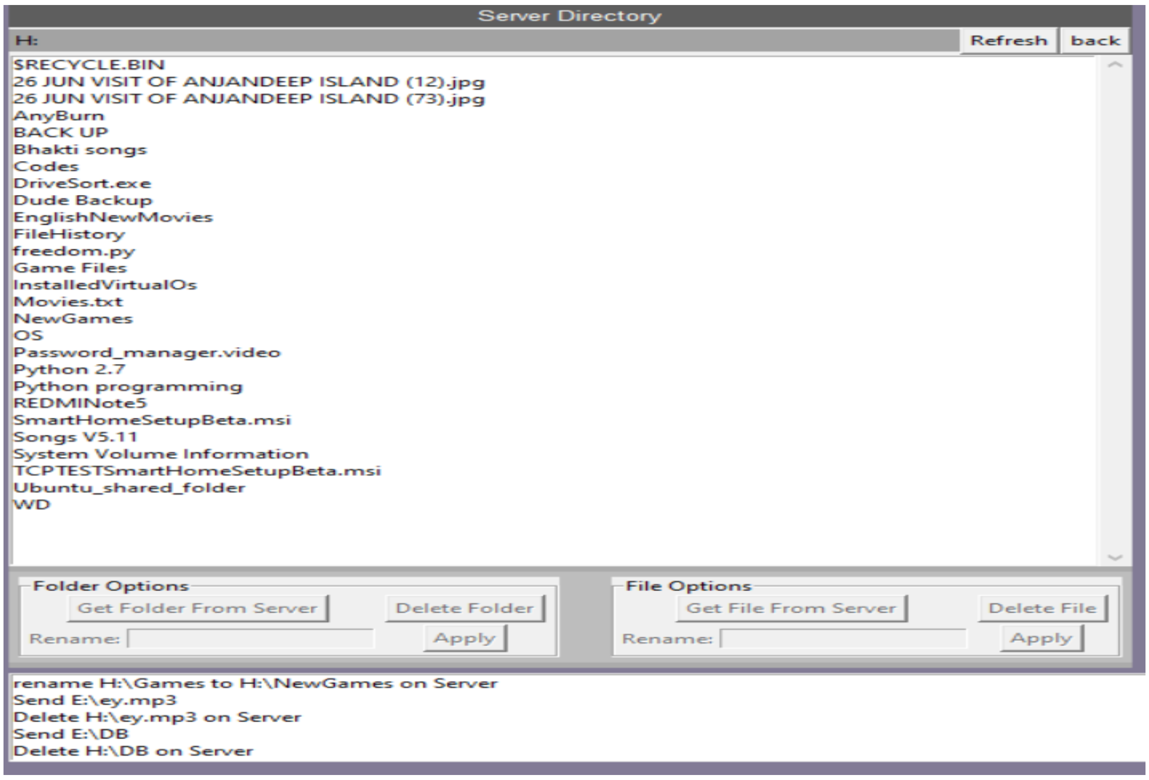
Select the folder to be deleted.



```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DB
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
InstalledVirtualOs
Movies.txt
NewGames
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

```
rename H:\NewUpdatedDriveSort.exe to H:\DriveSort.exe on Server
rename H:\Games to H:\NewGames on Server
Send E:\ey.mp3
Delete H:\ey.mp3 on Server
Send E:\DB
```

Now click on the delete folder button and the folder will be deleted shortly (shown below)

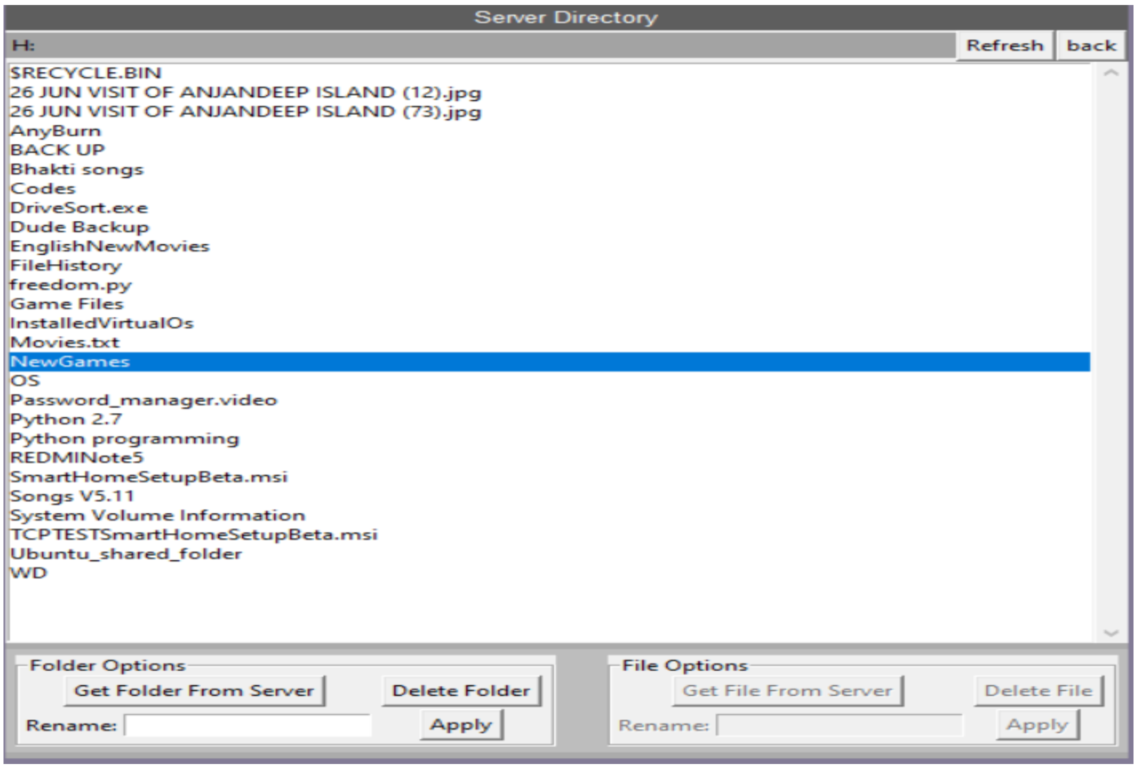


```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
InstalledVirtualOs
Movies.txt
NewGames
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

```
rename H:\Games to H:\NewGames on Server
Send E:\ey.mp3
Delete H:\ey.mp3 on Server
Send E:\DB
Delete H:\DB on Server
```

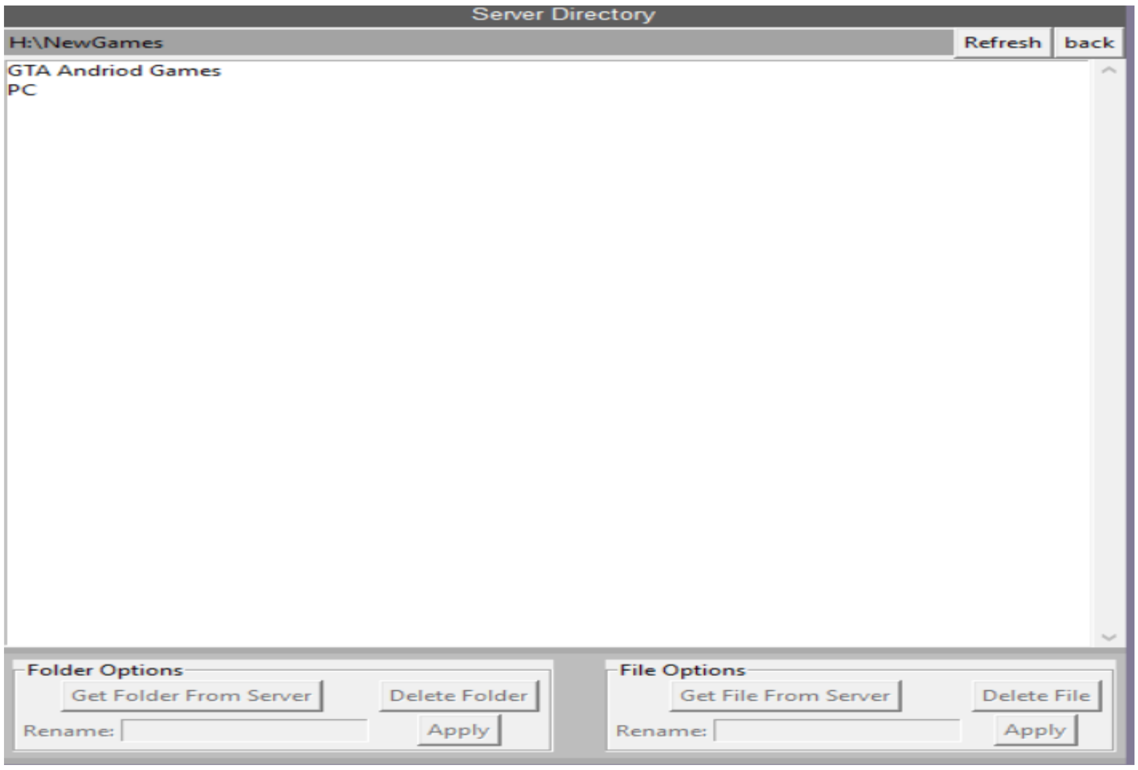
TO OPEN A SUBDIR:

Double click on the folder in the directory list.



```
$RECYCLE.BIN
26 JUN VISIT OF ANJANDEEP ISLAND (12).jpg
26 JUN VISIT OF ANJANDEEP ISLAND (73).jpg
AnyBurn
BACK UP
Bhakti songs
Codes
DriveSort.exe
Dude Backup
EnglishNewMovies
FileHistory
freedom.py
Game Files
InstalledVirtualOs
Movies.txt
NewGames
OS
Password_manager.video
Python 2.7
Python programming
REDMINote5
SmartHomeSetupBeta.msi
Songs V5.11
System Volume Information
TCPTESTSmartHomeSetupBeta.msi
Ubuntu_shared_folder
WD
```

The top directory label is updated and the directory list updates to views the contents of the subdirectory.(shown below)

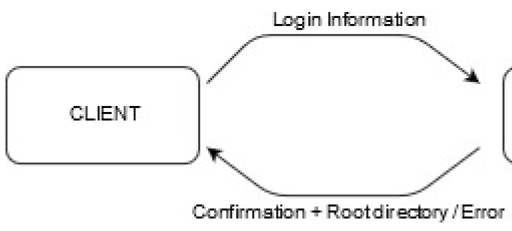


The bottom panel shows the log of all the actions done on the software.

If changes are made on server/client outside the MediaStream program, the refresh button helps the user to reflect the changes on the program.

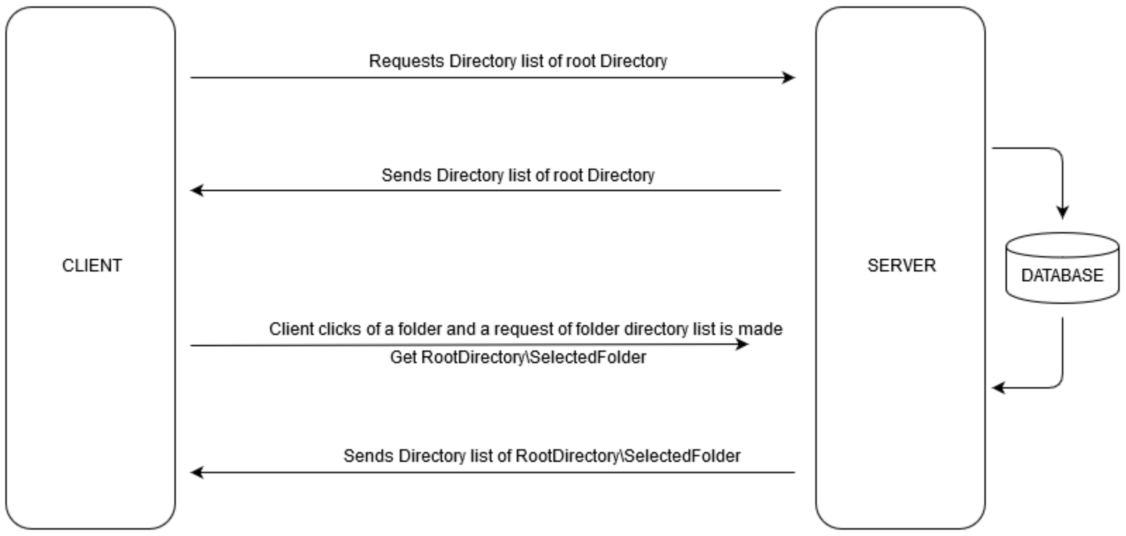
ALGORITHMS

CLIENT LOGIN:



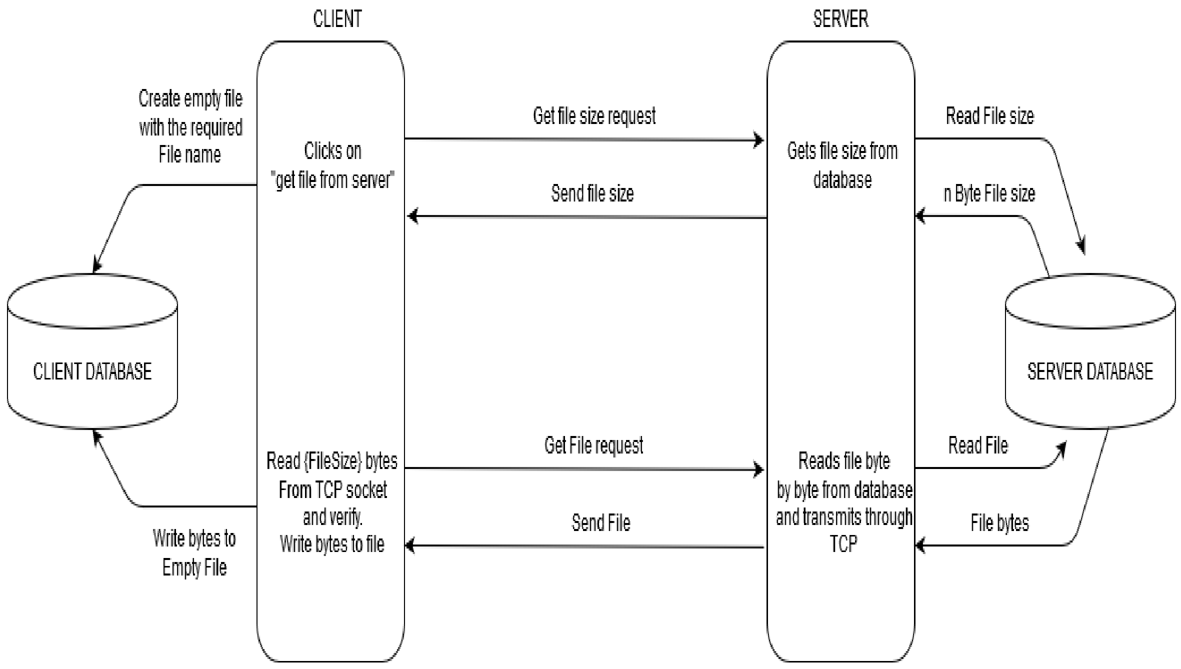
The login information contains server password only.

SETTING UP DIRECTORY LIST:


CLIENT
Requests Directory list of root Directory
Sends Directory list of root Directory
Client clicks of a folder and a request of folder directory list is made
Get RootDirectory\SelectedFolder
Sends Directory list of RootDirectory\SelectedFolder
SERVER
DATABASE

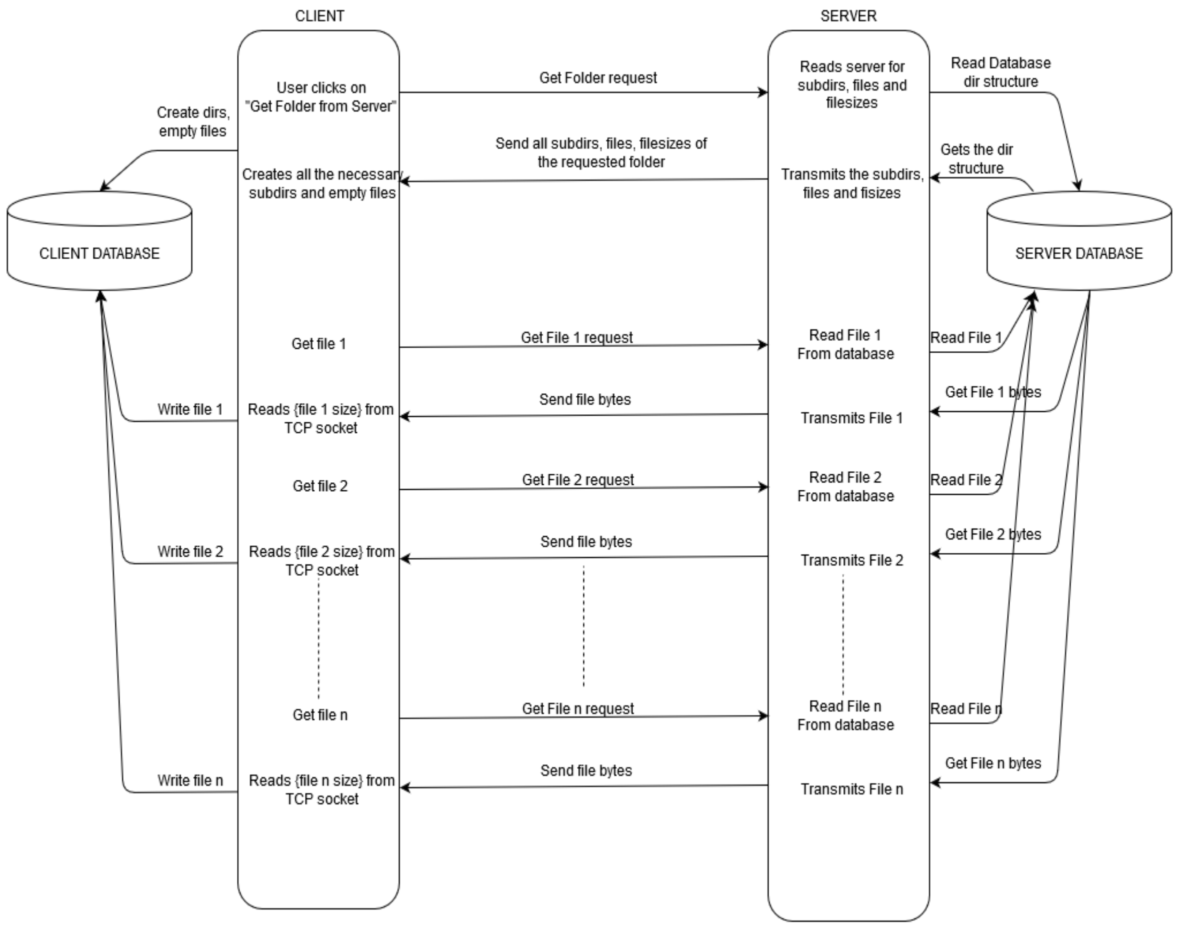
Whenever the client asks for the directory list of a folder, the server packs it in a format and transmits to the client. The client then unpacks the directory list and displays it on the directory listbox.

RECEIVE FILE FROM SERVER:


CLIENT
SERVER
Create empty file with the required File name
Clicks on "get file from server"
Get file size request
Send file size
Gets file size from database
Read File size
n Byte File size
CLIENT DATABASE
SERVER DATABASE
Read {FileSize} bytes From TCP socket and verify. Write bytes to file
Get File request
Send File
Reads file byte by byte from database and transmits through TCP
Read File
File bytes
Write bytes to Empty File

The client requests a file's data from the server. The server then sends the information regarding the file (file size, file name) to the client. The client then creates an empty file with the data received from the server. Next, the client sends the get file request. Now the server transmits the file byte by byte. The client reads the socket byte by byte till the received byte size equals the file size. The transmission is completed and the client updates its client directory list

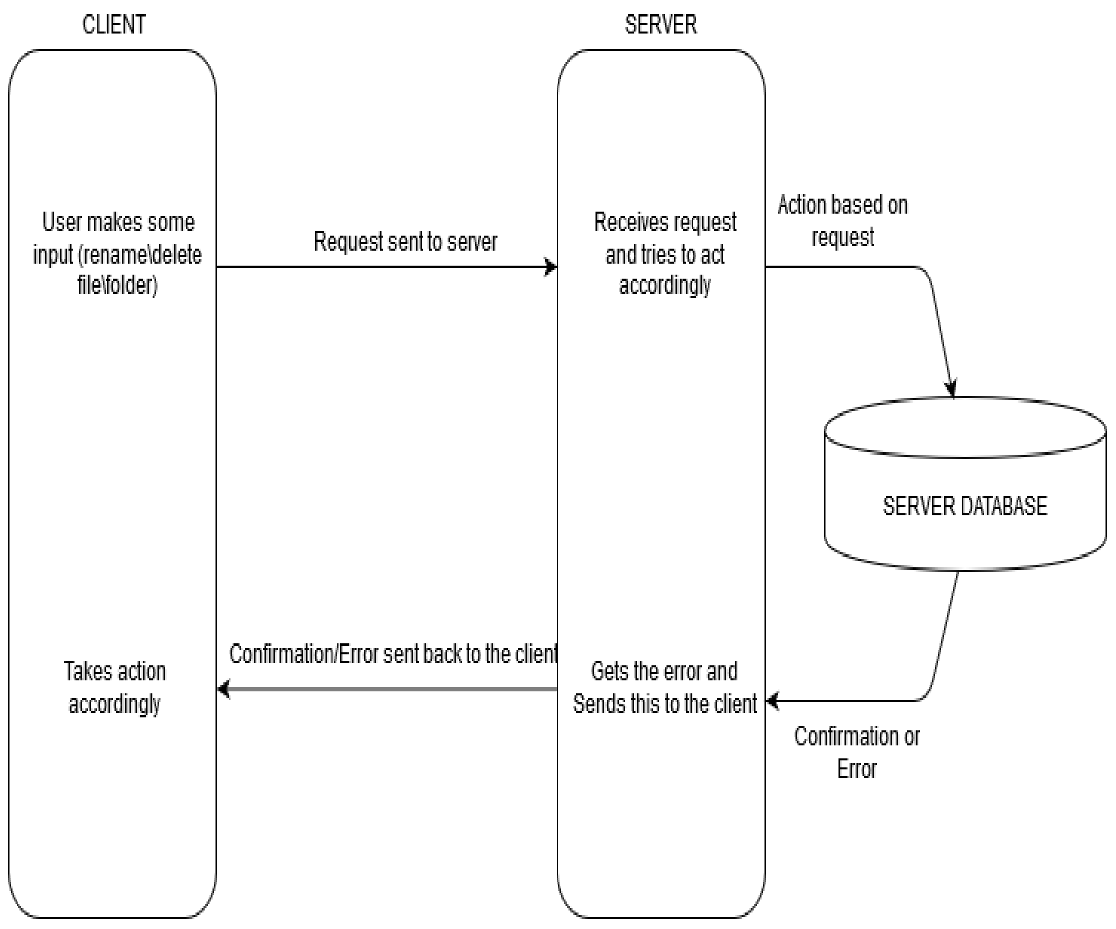
RECEIVE FOLDER FROM SERVER:



CLIENT
SERVER
User clicks on "Get Folder from Server"
Get Folder request
Reads server for subdirs, files and filesizes
Read Database dir structure
Create dirs, empty files
Send all subdirs, files, filesizes of the requested folder
Gets the dir structure
Creates all the necessary subdirs and empty files
Transmits the subdirs, files and fisizes
CLIENT DATABASE
SERVER DATABASE
Get file 1
Get File 1 request
Read File 1 From database
Read File 1
Write file 1
Reads {file 1 size} from TCP socket
Send file bytes
Transmits File 1
Get File 1 bytes
Get file 2
Get File 2 request
Read File 2 From database
Read File 2
Write file 2
Reads {file 2 size} from TCP socket
Send file bytes
Transmits File 2
Get File 2 bytes
Get file n
Get File n request
Read File n From database
Read File n
Write file n
Reads {file n size} from TCP socket
Send file bytes
Transmits File n
Get File n bytes

The client requests folder details to the server. The server fetch details from its database. This details fetch includes subdirectory list, files list and file sizes. The server then packs all these data in a format that the client understands and transmits this data. The client receives this data and unpacks it. The client then creates subdirectories, empty files according the data received from the server. Now, the client loops on all the files and makes file requests (the previous algorithm). This is done till the last file. In the end, the client updates its directory list to reflect the changes made.

MAKE CHANGES SUCH AS RENAMING OR DELETING:


CLIENT
SERVER
User makes some input (rename\delete file\folder)
Request sent to server
Receives request and tries to act accordingly
Action based on request
SERVER DATABASE
Confirmation/Error sent back to the client
Takes action accordingly
Gets the error and Sends this to the client
Confirmation or Error

The client requests the server to make required changes. If the changes are completed successfully, the server sends a confirmation back to the client. The client updates its server directory list. If the changes are not completed successfully, the server sends the error message to the client and the client shows this error message to the user.

FURTHER DEVELOPMENT

Just as nobody is perfect, no program is perfect. There is always scope for improvement. If you are an expert in python, try to read and understand the code. It is not hard as *simplicity* is Python's one of the strong points. After understanding the code, try to make changes in the algorithms to make it more efficient. After you've successfully mastered the code, try to customize the program.

SOME TIPS FOR IMPROVEMENT:

- Compression techniques can be used for faster and efficient transmission of data. This can affect the processing speed but the network congestion can be avoided.
- Encryption techniques can be used so that other people in the same network cannot interpret the data flowing between different nodes.
- Network equipment can be upgraded so the data transfer can be made at an even faster rate.

This code has been written in a flexible manner so that any changes can be easily made to the code to account for an individual's need.

CONCLUSION

The software MEDIA STREAM has been created in python on a windows machine with python 3.6.5 installed and is ready to be setup. The program has been tested and retested by our team mates to minimize bugs. The code is flexible and user friendly. A project report is created so as to understand the code better.

Teamwork, development process, debugging… were the goals achieved by the project.

The concept of master-slave networks has been strengthened in our teammates.

BIBLIOGRAPHY

- Class XI and Class XII text SUMITA ARORA about programming with python.
- www.tutorialspoint.com
- www.stackoverflow.com
- www.wikipedia.org