Joseph Salazar Proj1:P2 Dr. John

Diagram, schematic

Description automatically generated

client.py

'''

Created on Feb 17, 2023

@author: Joey Salazar

'''

import socket

import time

from DSMessage import DSMessage

from DSComm import DSComm

middleware = 50000

def listen\_for\_response(sock):

    comm = DSComm(sock)

    while True:

        mess = comm.recvMessage()

        if mess:

            tipe = mess.getType()

            data = mess.getData().decode('utf-8')

            print(data)

            if tipe == 'OKOK' or tipe == 'ERRO':

                print()

                break

def ClientProtocol(sock):

    msg = DSMessage()

    line = input('Enter Command: ')

    msg.setType(line[:4])

    d = line[4:]

    data = d.encode('utf8')

    msg.setData(data)

    comm = DSComm(sock)

    comm.sendMessage(msg)

    listen\_for\_response(sock)

if \_\_name\_\_ == '\_\_main\_\_':

    mainserv = socket.socket()

    mainserv.connect(('localhost', middleware))

    while True:

        ClientProtocol(mainserv)

    mainserv.close()

Main.py

'''

Created on Feb 17, 2023

@author: Joey Salazar

'''

import socket

import hashlib

import time

from DSMessage import DSMessage

from DSComm import DSComm

current\_user\_hash = ''

signed\_in = False

client = 50000

storage = 51000

def hash\_string(string\_to\_hash):

    hash\_obj = hashlib.sha256(string\_to\_hash.encode())

    hex\_string = hash\_obj.hexdigest()

    return hex\_string[:4]

def send\_data(data, type, sock):

    mess = DSMessage()

    mess.setType(type)

    mess.setData(data.encode('utf-8'))

    comm = DSComm(sock)

    comm.sendMessage(mess)

def receive\_data(dbsock):

    comm = DSComm(dbsock)

    while True:

        mess = comm.recvMessage()

        if mess:

            tipe = mess.getType()

            data = mess.getData().decode('utf-8')

            print(tipe,':',data)

            if tipe == 'OKOK' or tipe == 'ERRO':

                print()

                return tipe, data

def sign\_in\_success(credentials, clisock, dbsock):

    print('Login Success!')

    global current\_user\_hash

    current\_user\_hash = hash\_string(credentials)

    global signed\_in

    signed\_in = True

    print('Hash: ', current\_user\_hash)

    '''Set Globals so that program knows someone is signed in'''

    send\_data('Successful Login\nType \'MENU\' to display commands', 'OKOK', clisock)

    '''Send status to client'''

    '''Return to main protocol'''

    return

def login(data, clisock, dbsock):

    #print('Entered Login Protocol: ' + data)

    with open('userdb.txt', 'r') as file:

        for line in file:

            if data == line.rstrip():

                sign\_in\_success(data, clisock, dbsock)

                return

        send\_data('Error Signing In', 'ERRO', clisock)

def logout(data, clisock, dbsock):

    try:

        global current\_user\_hash

        current\_user\_hash = ''

        global signed\_in

        signed\_in = False

        send\_data('Successful Logout', 'OKOK', clisock)

        #Or break connection

    except:

        send\_data('Error Logging Out', 'ERRO', clisock)

    return

def menu(data, clisock, dbsock):

    with open('menu.txt', 'r') as file:

        contents = file.read()

        send\_data(contents, 'OKOK' ,clisock)

    return

def stor\_file(data, clisock, dbsock):

    '''

        1.)Send command to storage server along with file data

        2.)Receive filename and status back

        .) If responses OKOK

                add file to userDB

            else

                send client erro

        have not considered overwrite protocol yet

    '''

#try:

    global current\_user\_hash

    fname = current\_user\_hash + data

    send\_data(fname, 'STOR', dbsock)

    status, fname = receive\_data(dbsock)

    print('From DB\n\tStatus: ', status, '\t', 'fname: ', fname)

    if status == 'OKOK':                    #--------------> DATA or OKOK?

        with open('userdb.txt','a') as file:

            file.write('\n' + fname)

        send\_data('Stored File!', 'OKOK', clisock)

    else:

        send\_data('Couldn\'t store','ERRO',clisock )

#except:

    #send\_data('Couldn\'t communicate with dbserver', 'ERRO', clisock)

def retr\_file(fname,  clisock, dbsock):

    '''

        1.)Send command to storage server along with filename

        2.)Receive data and status back

        .) If responses OKOK

                send to client

            else

                send client erro

    '''

    global current\_user\_hash

    name = current\_user\_hash + fname

    with open('userdb.txt','r') as file:

        for line in file:

            if line[4:] == fname and line[:4] == current\_user\_hash:

                try:

                    send\_data(fname, 'RETR', dbsock)

                    status, data = receive\_data(dbsock)

                    if status == 'OKOK':

                        send\_data(data, 'OKOK', clisock)

                        return

                    else:

                        send\_data('Couldn\'t retreive file contents','ERRO',clisock )

                        return

                except:

                    send\_data('Couldn\'t retreive file contents','ERRO',clisock )

                    return

        send\_data('Couldn\'t find file','ERRO',clisock )

def dele\_file(fname, clisock, dbsock):

    global current\_user\_hash

    name = current\_user\_hash + fname

    with open('userdb.txt', 'r+') as file:

        lines = file.readlines()

        file.seek(0)  # move the file pointer back to the beginning

        found\_file = False

        for line in lines:

            if line[:4] == current\_user\_hash and line[4:].rstrip() == fname:

                try:

                    send\_data(name, 'DELE', dbsock)

                    status, data = receive\_data(dbsock)

                    if status == 'OKOK':

                        lines.remove(line)  # remove the line from the list

                        file.write(''.join(lines))  # write the updated list back to the file

                        file.truncate()  # truncate the remaining content

                        found\_file = True

                        send\_data('Successful Delete', 'OKOK', clisock)

                    else:

                        send\_data('Couldn\'t delete file contents', 'ERRO', clisock)

                    break

                except:

                    send\_data('Delete Exception', 'ERRO', clisock)

                    break

        if not found\_file:

            send\_data('File doesn\'t exist', 'ERRO', clisock)

def list\_files(data, clisock, dbsock):

    files = ''

    with open('userdb.txt','r') as file:

        for line in file:

            if current\_user\_hash in line:

                files += line[4:] + '\n'

    if files == '':

        send\_data('No files found/exist', 'ERRO', clisock)

    else:

        send\_data(files, 'OKOK', clisock)

def data\_in(data):

    pass

def okok\_in(data):

    pass

def erro\_in(data):

    pass

def decode\_command(line, data, clisock, dbsock = None):

    if signed\_in == False:

        if line != 'LGIN':

            send\_data('Use \'LGIN\' to login first!','ERRO',clisock )

        else:

            login(data, clisock, dbsock)

    else:

        if line == 'LGOT':

            logout(data, clisock, dbsock)

        elif line == 'MENU':

            menu(data, clisock, dbsock)

        elif line == 'STOR':

            stor\_file(data, clisock, dbsock)

        elif line == 'RETR':

            retr\_file(data, clisock, dbsock)

        elif line == 'DELE':

            dele\_file(data, clisock, dbsock)

        elif line == 'LIST':

            list\_files(data, clisock, dbsock)

        else:

            send\_data('Cant Decode','ERRO',clisock )

    return

def main\_protocol(clientsock, dbsock):

    comm = DSComm(clientsock)

    while True:

        print()

        print('Main Protocol')

        print('\tSigned in: ',signed\_in)

        mess = comm.recvMessage()

        if mess:

            tipe = mess.getType()

            data = mess.getData().decode('utf-8')

            print('\t','\tData/Command:',data)

            decode\_command(tipe,data, clientsock, dbsock)

if \_\_name\_\_ == '\_\_main\_\_':

    #Connect to DBserver

    dbserv = socket.socket()

    dbserv.connect(('localhost', storage))

    #Allow for Client Connection

    clientserv = socket.socket()

    clientserv.bind(("localhost", client))

    clientserv.listen(5)

    while True:

        print('Listening on ', client)

        clientsock, raddr = clientserv.accept()

        main\_protocol(clientsock, dbserv)

        clientsock.close()

    clientserv.close()

    dbserv.close()

DBserver.py

'''

Created on Feb 17, 2023

@author: Joey Salazar

'''

import socket

import time

from DSMessage import DSMessage

from DSComm import DSComm

middleware = 51000

# 'filename:file\_contents'

# assume string is decoded already

def decode\_file\_contents(string):

    filename, file\_content = string.split(":")

    size\_of\_content = len(file\_content)

    return filename, size\_of\_content, file\_content

    #filename = filename.strip("'")

    #file\_content = file\_content.strip("'")

def send\_data(data, type, sock):

    mess = DSMessage()

    mess.setType(type)

    mess.setData(data.encode('utf-8'))

    comm = DSComm(sock)

    comm.sendMessage(mess)

def check\_if\_overwrite():

    pass

def overwrite():

    pass

def stor\_file(data, midsock):

    try:

        fname, size, content = decode\_file\_contents(data)

    except:

        send\_data('File contents not formatted correctly', 'ERRO', midsock)

    """

    Fill protocol here

    if check\_if\_overwrite() == True:

        overwrite()

    else:

    """

    print('Filename: ', fname, '\t', 'Contents: ', content)

    send\_data(fname, 'OKOK', midsock)

def retr\_file(fname, midsock):

    try:

        send\_data('You called RETR', 'OKOK', midsock)

    except:

        send\_data('File contents not formatted correctly', 'ERRO', midsock)

    #Fill protocol here

def dele\_file(data, midsock):

    try:

        send\_data('You called DELE', 'OKOK', midsock)

    except:

        send\_data('File contents not formatted correctly', 'ERRO', midsock)

    #Fill protocol here

def decode\_cmd(cmd, data, midsock):

    if cmd == 'STOR':

        stor\_file(data, midsock)

    elif cmd == 'RETR':

        retr\_file(data, midsock)

    elif cmd == 'DELE':

        dele\_file(data, midsock)

    else:

        send\_data('Invalid Command', 'ERRO', midsock)

def db\_protocol(middlesock):

    comm = DSComm(middlesock)

    print('DB Protocol')

    while True:

        print('\tListening for message')

        mess = comm.recvMessage()

        if mess:

            tipe = mess.getType()

            data = mess.getData().decode('utf-8')

            print('Command:', tipe, '\t', 'Data: ', data)

            decode\_cmd(tipe, data, middlesock)

if \_\_name\_\_ == '\_\_main\_\_':

    '''Assume TCP Connections'''

    mainserv = socket.socket()

    mainserv.bind(("localhost", middleware))

    mainserv.listen(5)

    while True:

        print('Listening on ', middleware)

        middlesock, raddr = mainserv.accept()

        db\_protocol(middlesock)

        middlesock.close()

    mainserv.close()

Testing

Text

Description automatically generated