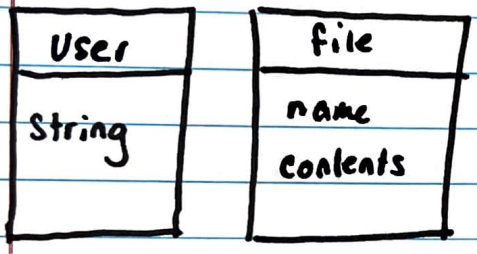
# Data Models

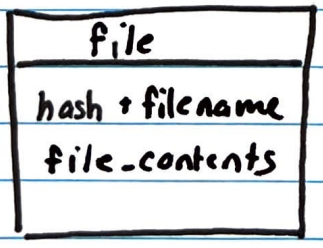
## Client :

- Login Ex. LGIN userpass
- Logout Ex. LGOT
- Store Ex. STORfile.txt : file_content
- Retrieve Ex. RETRfile.txt
- Delete Ex. DELEfile.txt
- Menu Ex. MENU
- List Ex. LIST → Requests commands from middleware




## Middle Ware :

- Same commands as client, but different protocol
- STOR, RETR, + DELE communicate w/ client + Dbservers
- LIST only communicates w/ client after checking local user db.



DB server :

| files |
| --- |
| hash + filename |
| file_contents |
| |

* Mod_time
* Crea_time
* etc...

- Only needs STOR, DELE, + RETR protocol
- Needs protocol for overwriting existing file

App Protocol :

Client -

Client Protocol (socket) → sends message
↳ listen (socket) → waits for response

Middleware -

MainProtocol (clisock, dbsock) → receive from client
↳ Decode (cmd, data, sock1, sock2) → decide what command

login (data, sock1, sock2)
↳ check if user exists, or fail

login_success (data, sock1, sock2)
↳ set globals + allow for more cmds

logout (data, sock1, sock2)
↳ reset globals + block commands

Menu (data, sock1, soc2)
↳ send menu to client

Middleware Cont. -

```
Stor_file (data, sock1, sock2)
    └ send to DB
        If ok:
            update local DB
            send ok to client
        Else
            send error to client

Retr_file (data, sock1, sock2)
    └ If in local DB:
            send req. to DB server
            If ok:
                send. to client
            Else
                send_error
        Else:
            send file_DNE error

DELE_file (Data, sock1, sock2)
    └ If in local DB:
            send_req to db server
            if okok:
                update local DB
                send ok to client
            else:
                send_error
        Else:
            send File DNE error

list_file ():
    get_files from local DB
    If None:
        send None to client
    Else:
        send files to client
```

App Protocol Cont:

DB Server:

db_Protocol (sock1, sock2)

↳ Receive from middleware
Then decode

Decode (data, sock1, sock2)

↳ call protocol or send_error
if cmd DNE

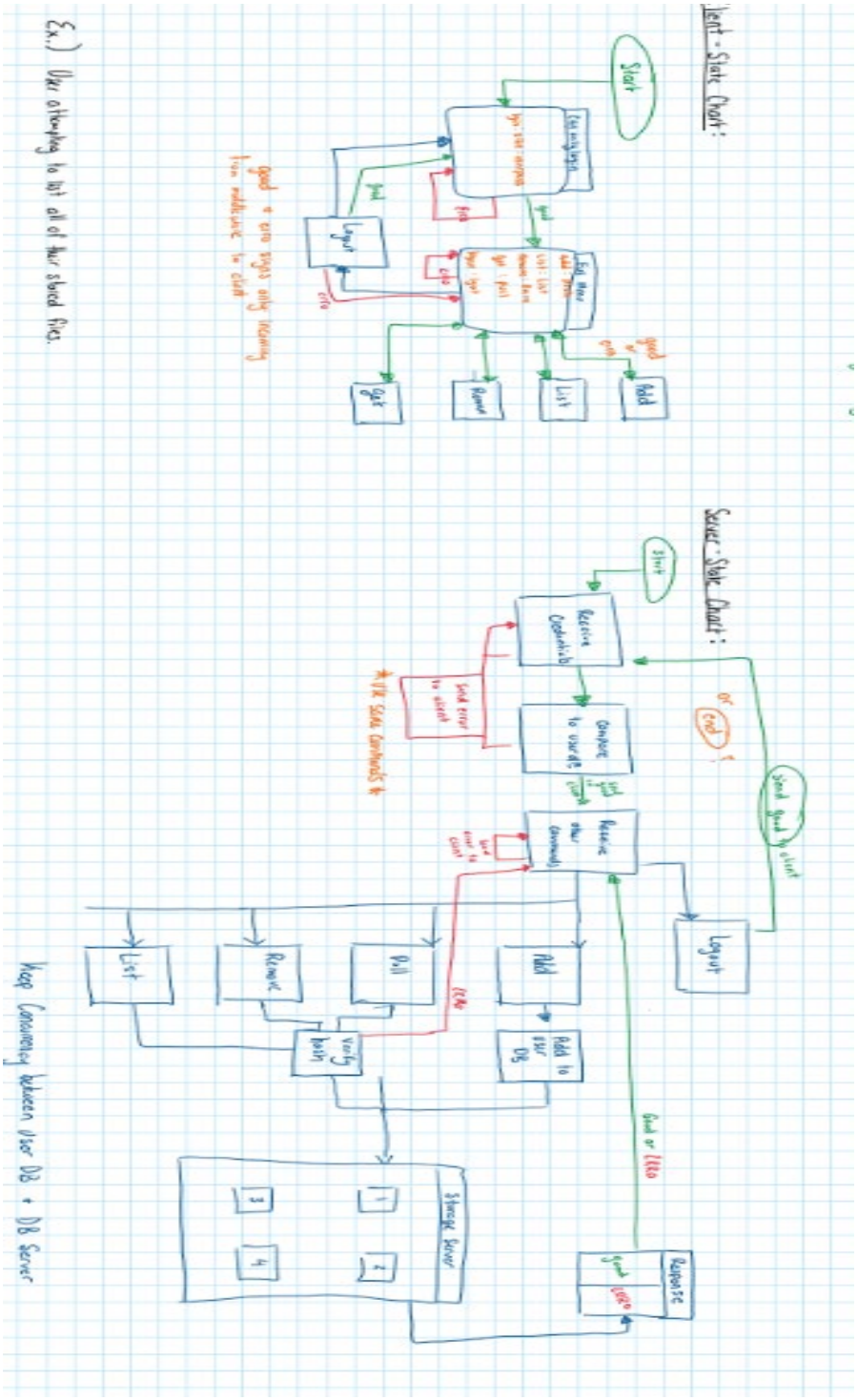
Still unsure depending on how + where we store on DBs 1-4

Dele ( )

Stor ( )
check_overwrite
overwrite

Retr ( )

Decode_file_contents (string)
returns file_name, file_content


Client - State Chart:
Start
Add
List
Remove
Exit
Server - State Chart:
Start
Receive Client Info
Send Error to client
Receive other Commands
Send Good to client
Logout
List
Remove
Add
Add to User DB
Verify Info
Send or Error
Shared Server
Ex.) User attempting to list all of their stored files.
Msgs Communicating between User DB + DB Server

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()
```

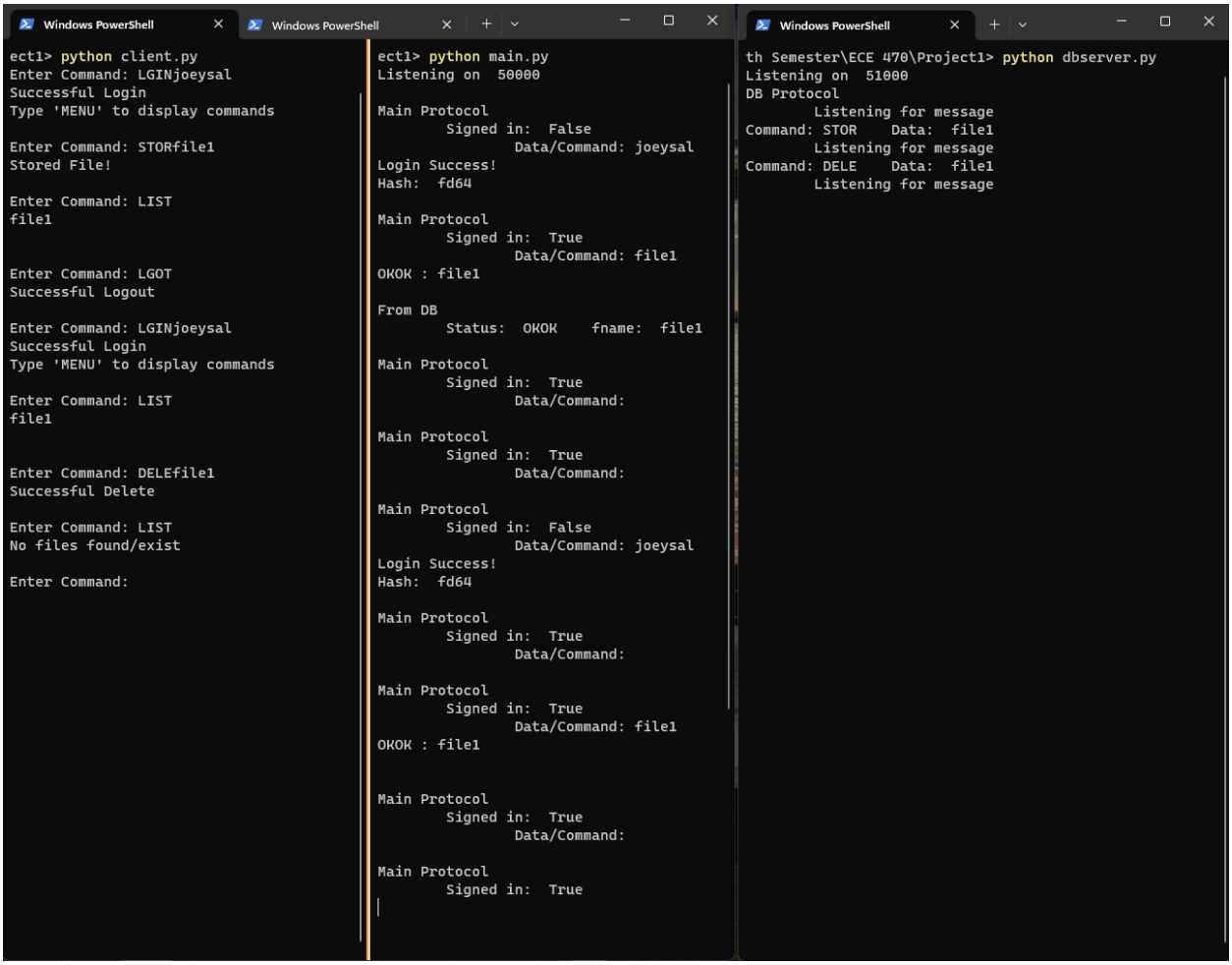
DBserver.py

```
'''
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


Windows PowerShell
Windows PowerShell
ect1> python client.py
Enter Command: LGINjoeysal
Successful Login
Type 'MENU' to display commands
Enter Command: STORfile1
Stored File!
Enter Command: LIST
file1
Enter Command: LGOT
Successful Logout
Enter Command: LGINjoeysal
Successful Login
Type 'MENU' to display commands
Enter Command: LIST
file1
Enter Command: DELEfile1
Successful Delete
Enter Command: LIST
No files found/exist
Enter Command:
ect1> python main.py
Listening on 50000
Main Protocol
Signed in: False
Data/Command: joeysal
Login Success!
Hash: fd64
Main Protocol
Signed in: True
Data/Command: file1
OKOK : file1
From DB
Status: OKOK fname: file1
Main Protocol
Signed in: True
Data/Command:
Main Protocol
Signed in: True
Data/Command:
Main Protocol
Signed in: False
Data/Command: joeysal
Login Success!
Hash: fd64
Main Protocol
Signed in: True
Data/Command:
Main Protocol
Signed in: True
Data/Command: file1
OKOK : file1
Main Protocol
Signed in: True
Data/Command:
Main Protocol
Signed in: True
Windows PowerShell
th Semester\ECE 470\Project1> python dbserver.py
Listening on 51000
DB Protocol
Listening for message
Command: STOR Data: file1
Listening for message
Command: DELE Data: file1
Listening for message