ReDI School of Digital Integration

Chatting Application on Local Network

Python (Hybrid) Final Project

Komal Chandrakant Bhapkar

We use tech to connect human potential and opportunity with dignity & humility

Want to Chat with your Friends ? Use my Chatting App ….

Tools & Concept

The project was created using Python along with the following libraries & Concepts:

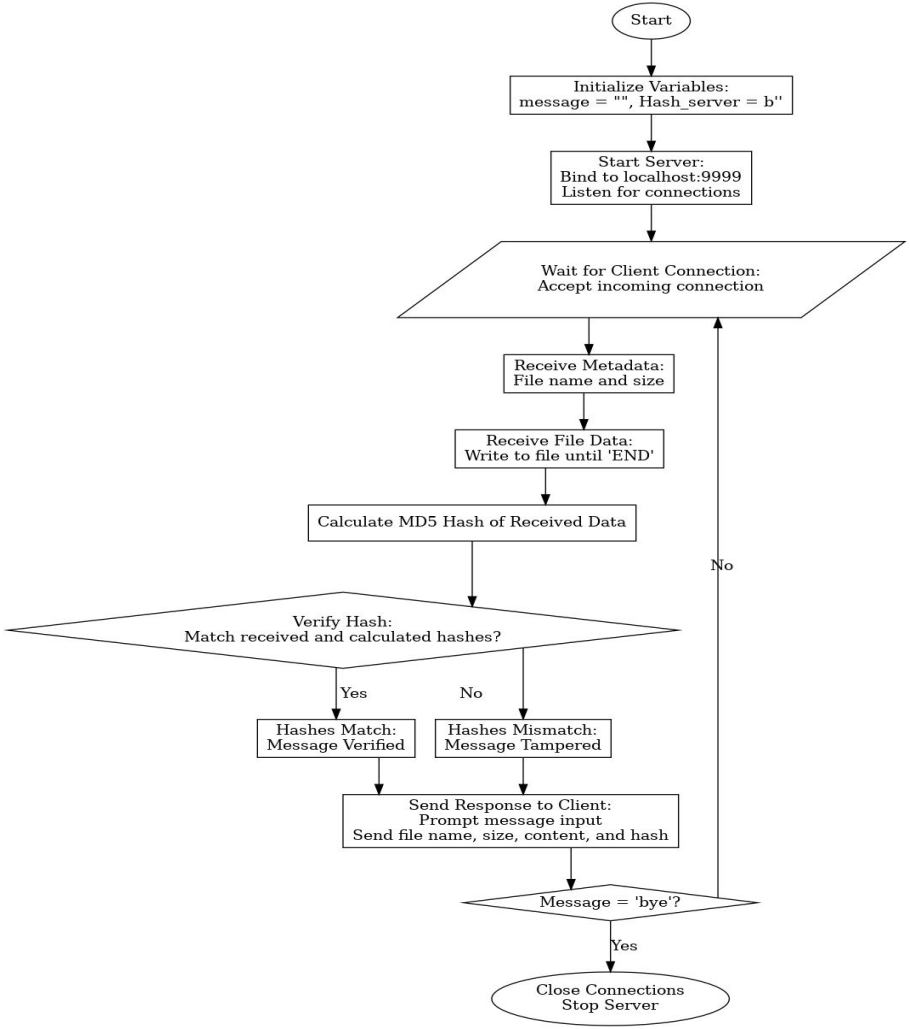
- **Socket**: For opening socket between users and transmitting the files
- **OS**: For extracting basename and File Size
- **Hashlib**: For calculating MD5 hash of message
- **Time**: For inserting delay for synchronisation between users
- **File Handling :** For reading and writing files
- **Exception Handling :** To avoid runtime errors
- **OPPs :** Hash Calculation is implemented using OPPs methods

I want to build a project which everyone can use in day to day life , my chatting app we can use easily across local network for easy & secure communication without internet. We can use it in classroom where mobile phones are not allowed. My Chatting apps helps me to explore Socket concept..

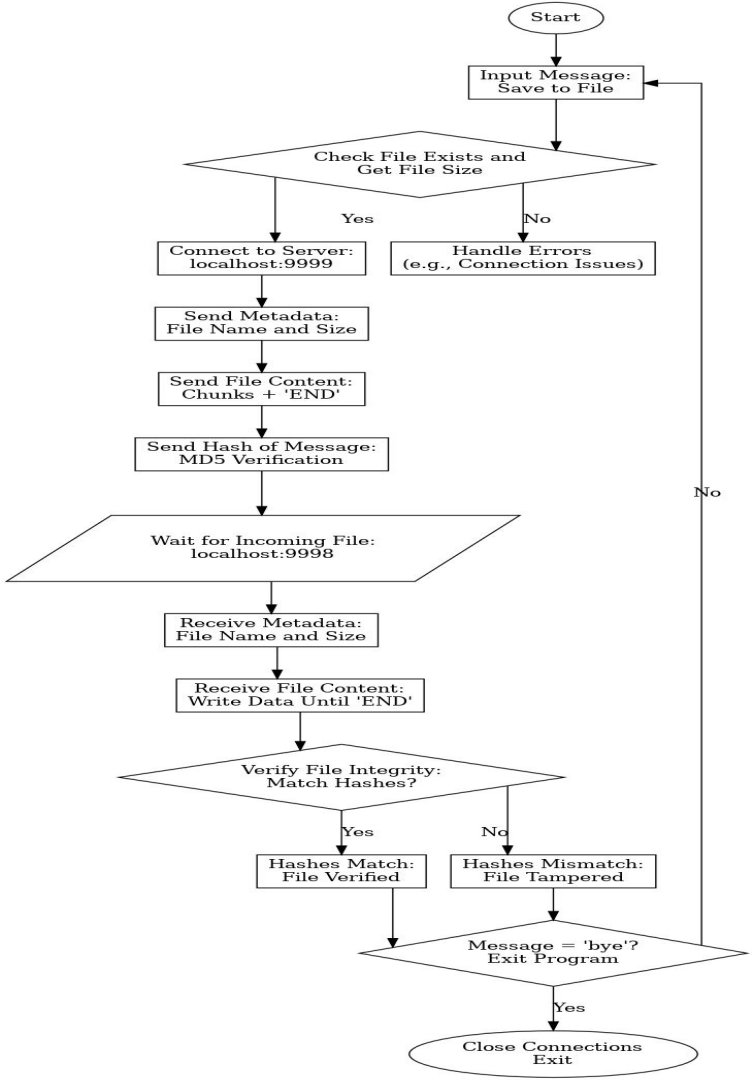
Overview of Chatting App

- There are two codes for two users Server and Client
- Both users are having their Text files and their socket for file transmission
- Server is opening the Socket & waiting for Client
- Client is connecting to the Server via Socket
- Client write message into its Text file & Calculate Hash of that message
- Client send its Text file to Server, after that it also send Hash
- Server receives the Text file from Client , write its content into its (Server's) Text file
- Server calculate Hash from Received message
- Server compares Calculated Hash with Received Hash , if it matches then it Read the Message, otherwise it prints 'Message is Tempered'
- To send message to Client ,Server follows same process as the Client
- Chatting get close if Server sends 'bye' message

Flow Chart of Server Code:



Flow Chart of Client Code:



Real World Implication

- Chat App use for Seamless chatting in local area network
- It provide secure communication, since messages are protected by MD5 Hash

Difficulties in building Chat App

- Sometimes length of the message ,actual message and hash of the message are getting append
- Synchronisation between users was a challenge socket connection was getting close before expectation
- I resolved them by adding sleep function of time library
- Socket closure was not handled by one user, so I implemented a patch of code which gives authority to Server python script to close the socket

Result

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER

```
PS A:\Work\ttt> python .\Server_1.py
Server is listening on port 9999...
End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Client :  Hey hii

write message For Client : hello
Sent file name: Text_From_ServerU2.txt
Sending file data...
Server is listening on port 9999...
End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Client :  How are you?

write message For Client : I am fine, How are you?
Sent file name: Text_From_ServerU2.txt
Sending file data...
Server is listening on port 9999...
End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Client :  I am also good , Thank you

write message For Client : Lets meet on monday
Sent file name: Text_From_ServerU2.txt
Sending file data...
Server is listening on port 9999...
End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Client :  okk then bye

write message For Client : bye
Sent file name: Text_From_ServerU2.txt
Sending file data...
PS A:\Work\ttt>
```

```
PS A:\Work\ttt> python .\Client_1.py
('localhost', 9999)

write message for Server  : Hey hii
Connected to server at ('localhost', 9999)
Sending file data...

End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Server :  hello

write message for Server  : How are you?
Connected to server at ('localhost', 9999)
Sending file data...

End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Server :  I am fine, How are you?

write message for Server  : I am also good , Thank you
Connected to server at ('localhost', 9999)
Sending file data...

End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Server :  Lets meet on monday

write message for Server  : okk then bye
Connected to server at ('localhost', 9999)
Sending file data...

End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Server :  bye
Exiting client.
Connection closed.
PS A:\Work\ttt>
```

Future Improvements

- Use the app for secure File transfer
- Add multiple users
- Enhance the App on WLAN

Server Code:

File Edit Selection View Go Run Terminal Help

Welcome | Server_1.py | Client_1.py

A: > Work > ttt > Server_1.py > ...

```python
import socket
import os
import hashlib
import time

# Server Code (User 2)

message = ""
Hash_server = b''

while message != "bye":

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('localhost', 9999))  #127.0.0.0
    server.listen(1)

    message = ""
    print("Server is listening on port 9999...")

    conn, addr = server.accept()

    #print(f"Connection established with {addr}")

    try:

        #file name
        file_name = conn.recv(1024).decode()
        print("Receiving file: {file_name}")
        #file size
        file_size = int(conn.recv(1024).decode())
        print(f"Expected file size: {file_size} bytes")

        # Open a file to write the received data
        with open("Received_" + os.path.basename(file_name), "wb") as file:
            received_size = 0
            while True:
                data = conn.recv(1024)

                if data == b"END":  # End of transmission
                    print("End of file transmission detected.")
                    break
                file.write(data)
                received_size += len(data)
                #Hash_server = hashlib.md5(data).hexdigest()
```

Welcome | Server_1.py | Client_1.py

A: > Work > ttt > Server_1.py > ...

```python
                file.write(data)
                received_size += len(data)
                #Hash_server = hashlib.md5(data).hexdigest()

                print(f"Received {received_size}/{file_size} bytes", end="\r")

        #time.sleep(1)
        Hash_From_sender_Client = conn.recv(1024)

        file_path = "Received_Messagefrom_U1.txt"
        File_1 = open(file_path,'r')

        message = File_1.read()

        Hash_server = hashlib.md5(message.encode()).hexdigest()

        if Hash_From_sender_Client.decode() == Hash_server:
            print("Received Message is successfully Hash Verified")
            print('Message Received from Client : ',message)
        else:
            print("Message is tampered")

        File_1.close()

        #print("\nHash of the message : ",Hash_server)
        #print("\nFile received successfully!")



    except Exception as e:
        print(f"An error occurred: {e}")
    except ValueError as ve:
        print(f"Error (Ex. Empty message): {ve}")
    finally:
        #conn.close()
        print(f" ")
        #server.close()
        #Socket is getting close outside of while loop

    ##########################################

    server_address_2 = ('localhost', 9998)

    message = input("write message For Client : ")
```

A: > Work > ttt > Server_1.py > ...

```python
    message = input("write message For Client : ")
    # Path to the file to be sent
    #file_path_send = "A:\\Work\\ttt\\Text_From_ServerU2.txt"
    file_path_send = "Text_From_ServerU2.txt"
    File_2 = open(file_path_send,'w')
    File_2.write(message)
    File_2.close()

    file_size = os.path.getsize(file_path_send)
    Server_send_to_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    Server_send_to_client.connect(server_address_2)

    Server_send_to_client.send(str(file_path_send).encode())
    print(f"Sent file name: {os.path.basename(file_path_send)}")

    file_size = os.path.getsize(file_path_send)

    # Send the file size
    Server_send_to_client.send(str(file_size).encode())

    time.sleep(0.2)

    with open(file_path_send, "rb") as file:
        print("Sending file data...")
        while chunk := file.read(1024):  # Read 1 KB chunks
            Server_send_to_client.send(chunk)

    # Send the end-of-file marker
    time.sleep(0.5)
    Server_send_to_client.send(b"END")


    #print("File sent successfully from server!")
    Hash_Ff = hashlib.md5(bytes(message.encode())).hexdigest()

    #time.sleep(1)
    Server_send_to_client.send(Hash_Ff.encode())


    #print(f"Socket is closed")

Server_send_to_client.close()
conn.close()
server.close()
```

Client Code:

Welcome | Server_1.py | Client_1.py

A: > Work > ttt > Client_1.py > ...

```python
import os
import socket
import hashlib
import time

#Client Code (User 1)
# Server details
server_address = ('localhost', 9999)


print(server_address)
message = ""

class HashHandler:
    """Class to handle hash calculation and verification."""

    def __init__(self, message=""):
        self.message = message

    def generate_hash(self):
        """Generate MD5 hash for the given message."""

        return hashlib.md5(self.message.encode()).hexdigest()

    def verify_hash(self, received_hash):
        """Verify if the received hash matches the hash of the message."""
        generated_hash = self.generate_hash()

        if generated_hash == received_hash.decode():
            return True
        return False

while True:
    print(' ')
    message = input("write message for Server  : ")
    # Path to the file to be sent
    #file_path = "A:\\Work\\ttt\\Messagefrom_U1.txt"
    file_path = "Messagefrom_U1.txt"
    File_1 = open(file_path,'w')
    File_1.write(message)
    File_1.close()

    try:
        # Check if the file exists
        if not os.path.exists(file_path):
```

A: > Work > ttt > Client_1.py > ...

```python
    try:
        # Check if the file exists
        if not os.path.exists(file_path):
            raise FileNotFoundError(f"File not found: {file_path}")

        # Get the file size
        file_size = os.path.getsize(file_path)

        # Create a TCP client socket
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        # Connect to the server
        client.connect(server_address)
        print(f"Connected to server at {server_address}")

        # Send the file name
        client.send(os.path.basename(file_path).encode())
        #print(f"Sent file name: {os.path.basename(file_path)}")

        time.sleep(0.1)
        # Send the file size
        client.send(str(file_size).encode())
        #print(f"Sent file size: {file_size} bytes")

        time.sleep(0.2)

        # Open the file and send its contents in chunks
        with open(file_path, "rb") as file:
            print("Sending file data...")
            while chunk := file.read(1024):  # Read 1 KB chunks
                client.send(chunk)

        # Send the end-of-file marker
        time.sleep(0.5)
        client.send(b"END")
        #print("File sent successfully!")
        hash_handler = HashHandler(message)
        hash_from_sender_client = hash_handler.generate_hash()  # Generate hash from client message
        client.send(hash_from_sender_client.encode())
        #print("The sender hex is------ ", hashlib.md5(bytes(message.encode())).hexdigest())
        #print("The sender hex is after and ------ ",( int(Hash_From_sender_Client.encode()) & int(key.encode())))


    except ConnectionRefusedError:
        print("Could not connect to the server. Is it running and listening on the correct port?")
```

Restricted Mode ⊗ 0 ⚠ 0 0 Live Share Ln 37, Col 32 (3 selected) Spaces: 4 UTF-8 CRLF Python

Welcome | Server_1.py | Client_1.py

A: > Work > ttt > Client_1.py > ...

```python

    except ConnectionRefusedError:
        print("Could not connect to the server. Is it running and listening on the correct port?")
    except FileNotFoundError as e:
        print(e)
    except Exception as e:
        print(f"An error occurred: {e}")
    finally:
        # Close the connection
        #client.close()
        print(" ")
        #Connection closed out side of the while loop.

##############################################################################

    server_send = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_send.bind(('localhost', 9998))
    server_send.listen(2)

    conn_send, addr_send = server_send.accept()

    file_name = conn_send.recv(1024).decode()
    #print(f"Receiving file: {file_name}")


    # Receive the file size
    file_size = int(conn_send.recv(1024).decode())
    #print(f"Expected file size: {file_size} bytes")


    # Open a file to write the received data
    with open("Received_" + os.path.basename(file_name), "wb") as file:
        received_size = 0
        while True:
            data = conn_send.recv(1024)
            if data == b"END":  # End of transmission
                print("End of file transmission detected.")
                break
            file.write(data)
            received_size += len(data)
            Hash_Client = hashlib.md5(data).hexdigest()
            print(f"Received {received_size}/{file_size} bytes", end="\r")

    #time.sleep(0.5)
```

File Edit Selection View Go Run Terminal Help

Welcome | Server_1.py | Client_1.py

A: > Work > ttt > Client_1.py > ...

```python
                print("End of file transmission detected.")
                break
            file.write(data)
            received_size += len(data)
            Hash_Client = hashlib.md5(data).hexdigest()
            print(f"Received {received_size}/{file_size} bytes", end="\r")

    #time.sleep(0.5)
    Hash_From_sender_Server = conn_send.recv(1024)#######

    file_path = "Received_Text_From_ServerU2.txt"
    File_3 = open(file_path,'r')


    message = File_3.read()

    hash_handler = HashHandler(message)  # Use message from file to verify

    #print('Received message is ', message)
    #print('Received Hash is ',Hash_From_sender_Server)

    if hash_handler.verify_hash(Hash_From_sender_Server):
        print("Received Message is successfully Hash Verified")
        print('Message Received from Server : ',message)
    else:
        print("Message is tampered")
    #print(message)
    File_3.close()

    server_send.close()
    conn_send.close()

    if message == 'bye':
        time.sleep(1)
        print("Exiting client.")
        break


client.close()
conn_send.close()
server_send.close()
print("Connection closed.")
```

Thank You ...!!!