

Chatting Application on Local Network

Python (Hybrid) Final Project

Komal Chandrakant Bhapkar



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

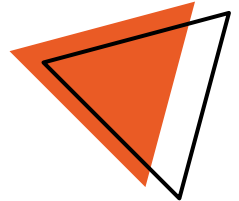
**Want to Chat with your Friends ? Use my
Chatting App**



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

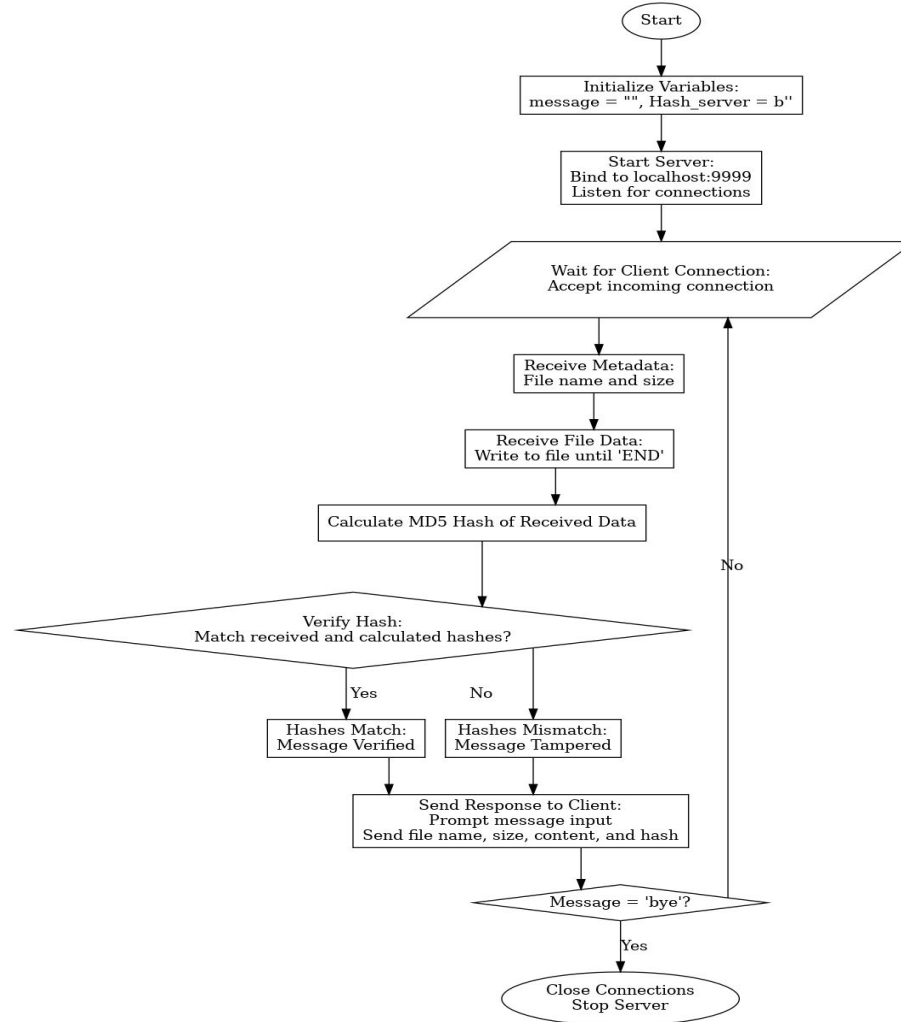


I want to build a project which everyone can use in day to day life , my chatting app we can use easily across local network for easy & secure communication without internet. We can use it in classroom where mobile phones are not allowed. My Chatting apps helps me to explore Socket concept..

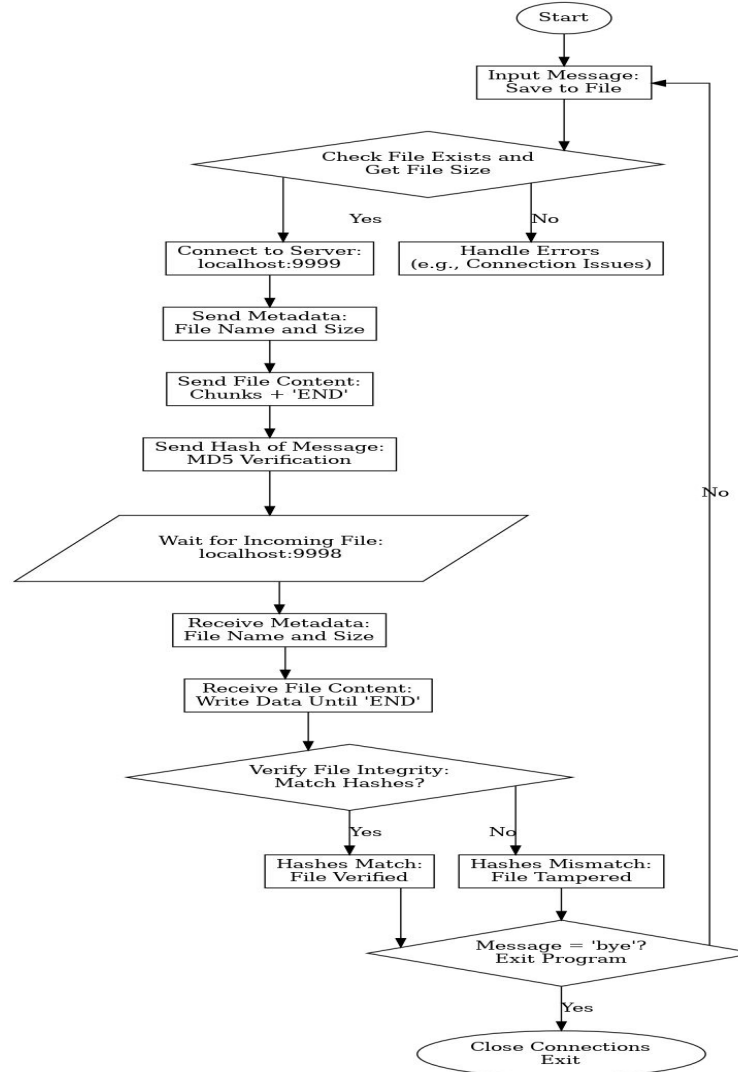
Overview of Chatting App

- 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...

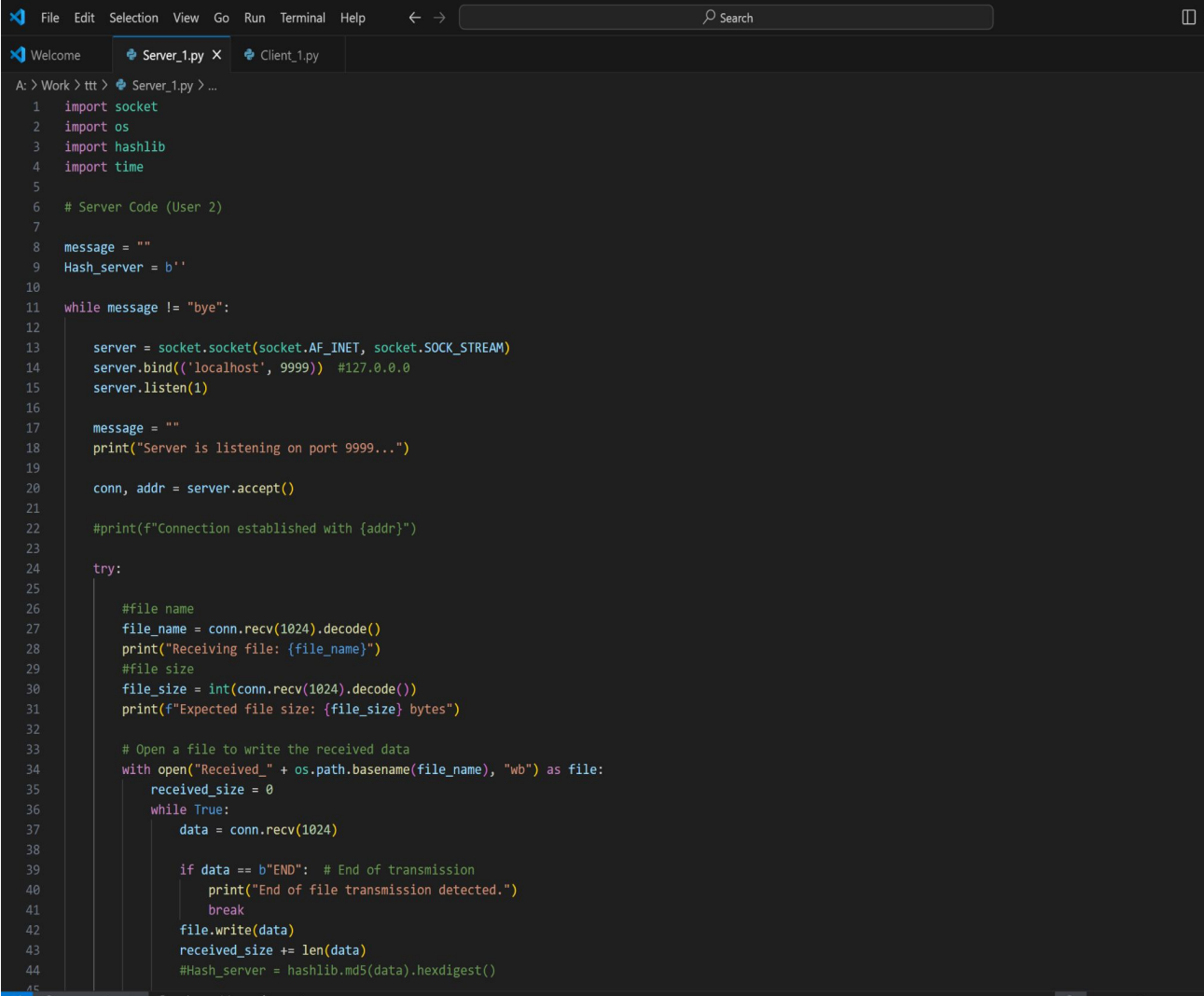
End of file transmission detected.
Received Message is successfully Hash Verified
Message Received from Server :  bye
Exiting client.
Connection closed.
PS A:\Work\ttt>
```

Future Improvements



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

Server Code:

A screenshot of a code editor window with a dark theme. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. Below the menu bar, there are tabs for 'Welcome', 'Server_1.py' (which is active), and 'Client_1.py'. The main area shows Python code for a server. The code starts with imports for socket, os, hashlib, and time. It then defines a message and a hash_server variable. A while loop runs as long as message is not 'bye'. Inside the loop, it creates a server socket, binds it to localhost on port 9999, and listens for connections. When a connection is accepted, it prints the address and enters a try block. In the try block, it receives the file name and size, opens a file to write the data, and then receives the data in chunks until 'END' is received. Finally, it calculates the MD5 hash of the data.

```
File Edit Selection View Go Run Terminal Help
Welcome Server_1.py Client_1.py
A: > Work > ttt > Server_1.py > ...
1 import socket
2 import os
3 import hashlib
4 import time
5
6 # Server Code (User 2)
7
8 message = ""
9 Hash_server = b''
10
11 while message != "bye":
12
13     server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14     server.bind(('localhost', 9999)) #127.0.0.0
15     server.listen(1)
16
17     message = ""
18     print("Server is listening on port 9999...")
19
20     conn, addr = server.accept()
21
22     #print(f"Connection established with {addr}")
23
24     try:
25
26         #file name
27         file_name = conn.recv(1024).decode()
28         print("Receiving file: {file_name}")
29         #file size
30         file_size = int(conn.recv(1024).decode())
31         print(f"Expected file size: {file_size} bytes")
32
33         # Open a file to write the received data
34         with open("Received_" + os.path.basename(file_name), "wb") as file:
35             received_size = 0
36             while True:
37                 data = conn.recv(1024)
38
39                 if data == b"END": # End of transmission
40                     print("End of file transmission detected.")
41                     break
42                 file.write(data)
43                 received_size += len(data)
44                 #Hash_server = hashlib.md5(data).hexdigest()
45
```

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

```
42         file.write(data)
43         received_size += len(data)
44         #Hash_server = hashlib.md5(data).hexdigest()
45
46         print(f"Received {received_size}/{file_size} bytes", end="\r")
47
48     #time.sleep(1)
49     Hash_From_sender_Client = conn.recv(1024)
50
51     file_path = "Received_Messagefrom_U1.txt"
52     File_1 = open(file_path, 'r')
53
54     message = File_1.read()
55
56     Hash_server = hashlib.md5(message.encode()).hexdigest()
57
58     if Hash_From_sender_Client.decode() == Hash_server:
59         print("Received Message is successfully Hash Verified")
60         print('Message Received from Client : ', message)
61     else:
62         print("Message is tampered")
63
64     File_1.close()
65
66     #print("\nHash of the message : ", Hash_server)
67     #print("\nFile received successfully!")
68
69
70
71 except Exception as e:
72     print(f"An error occurred: {e}")
73 except ValueError as ve:
74     print(f"Error (Ex. Empty message): {ve}")
75 finally:
76     #conn.close()
77     print(f" ")
78     #server.close()
79     #Socket is getting close outside of while loop
80
81 #####
82
83 server_address_2 = ('localhost', 9998)
84
85 message = input("write message For Client : ")
```

```
85     message = input("write message For Client : ")
86     # Path to the file to be sent
87     file_path_send = "A:\\Work\\ttt\\Text_From_ServerU2.txt"
88     file_path_send = "Text_From_ServerU2.txt"
89     File_2 = open(file_path_send, 'w')
90     File_2.write(message)
91     File_2.close()
92
93     file_size = os.path.getsize(file_path_send)
94     Server_send_to_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
95     Server_send_to_client.connect(server_address_2)
96
97     Server_send_to_client.send(str(file_path_send).encode())
98     print(f"Sent file name: {os.path.basename(file_path_send)}")
99
100    file_size = os.path.getsize(file_path_send)
101
102    # Send the file size
103    Server_send_to_client.send(str(file_size).encode())
104
105    time.sleep(0.2)
106
107    with open(file_path_send, "rb") as file:
108        print("Sending file data...")
109        while chunk := file.read(1024): # Read 1 KB chunks
110            Server_send_to_client.send(chunk)
111
112    # Send the end-of-file marker
113    time.sleep(0.5)
114    Server_send_to_client.send(b"END")
115
116
117    #print("File sent successfully from server!")
118    Hash_Ff = hashlib.md5(bytes(message.encode())).hexdigest()
119
120    #time.sleep(1)
121    Server_send_to_client.send(Hash_Ff.encode())
122
123
124    #print(f"Socket is closed")
125
126    Server_send_to_client.close()
127    conn.close()
128    server.close()
```

Client Code:

```
Welcome X Server_1.py Client_1.py X
A: > Work > ttt > Client_1.py > ...
1 import os
2 import socket
3 import hashlib
4 import time
5
6 #Client Code (User 1)
7 # Server details
8 server_address = ('localhost', 9999)
9
10
11 print(server_address)
12 message = ""
13
14 class HashHandler:
15     """Class to handle hash calculation and verification."""
16
17     def __init__(self, message=""):
18         self.message = message
19
20     def generate_hash(self):
21         """Generate MD5 hash for the given message."""
22
23         return hashlib.md5(self.message.encode()).hexdigest()
24
25     def verify_hash(self, received_hash):
26         """Verify if the received hash matches the hash of the message."""
27         generated_hash = self.generate_hash()
28
29         if generated_hash == received_hash.decode():
30             return True
31         return False
32
33 while True:
34     print(' ')
35     message = input("write message for Server : ")
36     # Path to the file to be sent
37     #file_path = "A:\\Work\\ttt\\Messagefrom_U1.txt"
38     file_path = "Messagefrom_U1.txt"
39     file_1 = open(file_path, 'w')
40     file_1.write(message)
41     file_1.close()
42
43     try:
44         # Check if the file exists
45         if not os.path.exists(file_path):
```

```

43     try:
44         # Check if the file exists
45         if not os.path.exists(file_path):
46             raise FileNotFoundError(f"File not found: {file_path}")
47
48         # Get the file size
49         file_size = os.path.getsize(file_path)
50
51         # Create a TCP client socket
52         client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
53
54         # Connect to the server
55         client.connect(server_address)
56         print(f"Connected to server at {server_address}")
57
58         # Send the file name
59         client.send(os.path.basename(file_path).encode())
60         print(f"Sent file name: {os.path.basename(file_path)}")
61
62         time.sleep(0.1)
63         # Send the file size
64         client.send(str(file_size).encode())
65         print(f"Sent file size: {file_size} bytes")
66
67         time.sleep(0.2)
68
69         # Open the file and send its contents in chunks
70         with open(file_path, "rb") as file:
71             print("Sending file data...")
72             while chunk := file.read(1024): # Read 1 KB chunks
73                 client.send(chunk)
74
75         # Send the end-of-file marker
76         time.sleep(0.5)
77         client.send(b"END")
78         print("File sent successfully!")
79         hash_handler = HashHandler(message)
80         hash_from_sender_client = hash_handler.generate_hash() # Generate hash from client message
81         client.send(hash_from_sender_client.encode())
82         print("The sender hex is----- ", hashlib.md5(bytes(message.encode())).hexdigest())
83         print("The sender hex is after and ----- ", (int(hash_from_sender_client.encode()) & int(key.encode()))
84
85
86     except ConnectionRefusedError:
87         print("Could not connect to the server. Is it running and listening on the correct port?")

```


Client_1.py X

Server_1.py X

Welcome

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

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

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
Search

Welcome Server_1.py Client_1.py X

A: > Work > ttt > Client_1.py ...
121         print("End of file transmission detected.")
122         break
123         file.write(data)
124         received_size += len(data)
125         Hash_Client = hashlib.md5(data).hexdigest()
126         print(f"Received {received_size}/{file_size} bytes", end="\n")
127
128         #time.sleep(0.5)
129         Hash_From_sender_Server = conn_send.recv(1024)#####
130
131         file_path = "Received_Text_From_ServerU2.txt"
132         File_3 = open(file_path,'r')
133
134
135         message = File_3.read()
136
137         hash_handler = HashHandler(message) # Use message from file to verify
138
139         #print('Received message is ', message)
140         #print('Received Hash is ',Hash_From_sender_Server)
141
142         if hash_handler.verify_hash(Hash_From_sender_Server):
143             print("Received Message is successfully Hash Verified")
144             print('Message Received from Server : ',message)
145         else:
146             print("Message is tampered")
147         #print(message)
148         File_3.close()
149
150         server_send.close()
151         conn_send.close()
152
153         if message == 'bye':
154             time.sleep(1)
155             print("Exiting client.")
156             break
157
158
159         client.close()
160         conn_send.close()
161         server_send.close()
162         print("Connection closed.")
```





Thank You!!!

