Sanjana Asrani
D17B/01/Batch B

Lab-02: Server, client, bidirectional streaming
Extension: Asynchronous and synchronous

gRPC is a stateless Transient (Both client and server need to be on together) communication protocol.
gRPC, or Remote Procedure Call, is an open-source remote procedure call (RPC) framework developed by Google. It uses Protocol Buffers (protobuf) as its interface definition language. gRPC enables efficient and robust communication between distributed systems and is often used for building microservices, APIs, and other networked applications.

The code below demonstrates the use of gRPC (Google Remote Procedure Call), a high-performance open-source RPC (Remote Procedure Call) framework that uses HTTP/2 for transport and Protocol Buffers as the interface definition language. It enables efficient and reliable communication between distributed systems.

Server Side:
- The server uses a thread pool ('futures.ThreadPoolExecutor') to handle incoming requests concurrently.

1. 'GreeterServicer' Class:
   - Implements the service defined in 'greet_pb2_grpc.GreeterServicer'.
   - Defines two RPC methods:
     - 'ChattyClientSaysHello': Accepts a stream of requests and responds with a delayed reply.
     - 'InteractingHello': Accepts a stream of requests and responds asynchronously.

2. Server Setup:
   - Uses 'grpc.server' with a 'ThreadPoolExecutor' to handle concurrent requests.
   - Adds the 'GreeterServicer' to the server.
   - Starts the server on an insecure port ('localhost:50051').
   - Uses 'server.wait_for_termination()' to keep the server running.

Client Side:
1. Request Protobuf Definitions ('greet_pb2'):
   - Defines message structures used for communication, such as 'HelloRequest', 'HelloReply', and 'DelayedReply'.

2. Service Stub ('greet_pb2_grpc.GreeterStub'):
   - Generated by the gRPC tools from the service definition ('greet_pb2_grpc.GreeterServicer').
   - Used to make RPC calls to the server.

3. 'get_client_stream_requests' Function:
   - An asynchronous generator function that takes user input and yields 'HelloRequest' messages in a loop.

   - Simulates a client that sends requests with a delay of 5 seconds.

4. Client Execution ('run_synchronous' and 'run_asynchronous' Functions):
   - Establishes an insecure gRPC channel with the server ('grpc.aio.insecure_channel').
   - Creates a service stub ('greet_pb2_grpc.GreeterStub') using the channel.

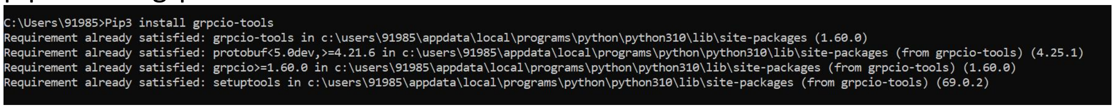
   - 'run_synchronous' Function:
     - Calls 'ChattyClientSaysHello' with a stream of requests and prints the delayed response.

   - 'run_asynchronous' Function:
     - Calls 'InteractingHello' asynchronously with multiple concurrent requests.
     - Prints responses as they are received.

   - 'run' Function:
     - Allows the user to choose between synchronous and asynchronous communication.

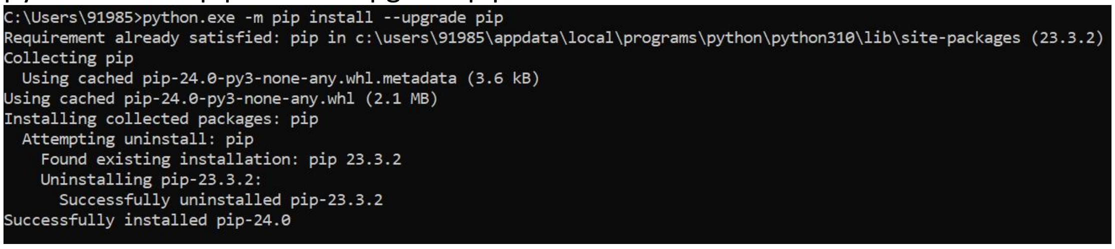
<u>Steps:</u>

pip3 install grpcio-tools



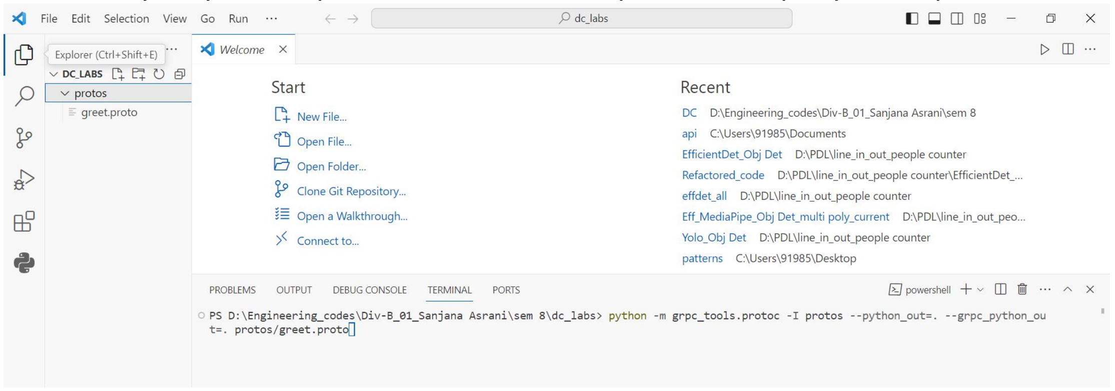
```
C:\Users\91985>Pip3 install grpcio-tools
Requirement already satisfied: grpcio-tools in c:\users\91985\appdata\local\programs\python\python310\lib\site-packages (1.60.0)
Requirement already satisfied: protobuf<5.0dev,>=4.21.6 in c:\users\91985\appdata\local\programs\python\python310\lib\site-packages (from grpcio-tools) (4.25.1)
Requirement already satisfied: grpcio>=1.60.0 in c:\users\91985\appdata\local\programs\python\python310\lib\site-packages (from grpcio-tools) (1.60.0)
Requirement already satisfied: setuptools in c:\users\91985\appdata\local\programs\python\python310\lib\site-packages (from grpcio-tools) (69.0.2)
```

python.exe -m pip install --upgrade pip



```
C:\Users\91985>python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\91985\appdata\local\programs\python\python310\lib\site-packages (23.3.2)
Collecting pip
  Using cached pip-24.0-py3-none-any.whl.metadata (3.6 kB)
Using cached pip-24.0-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.3.2
    Uninstalling pip-23.3.2:
      Successfully uninstalled pip-23.3.2
Successfully installed pip-24.0
```

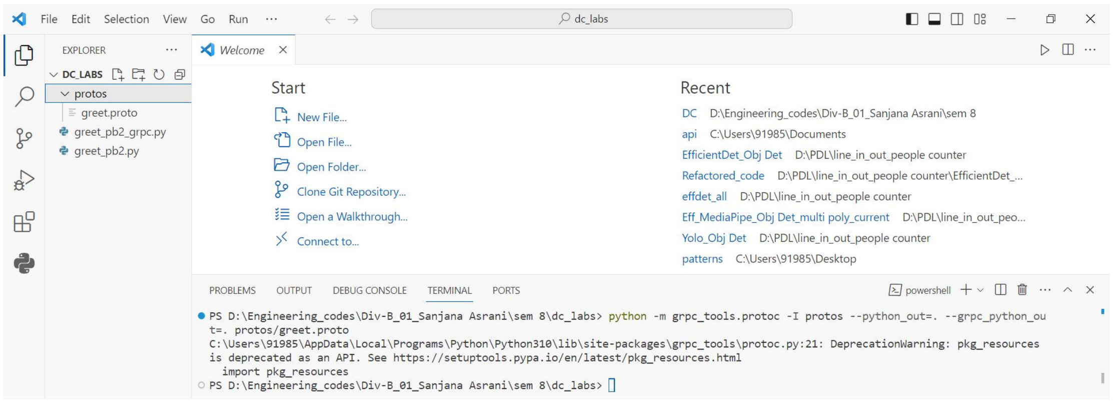
You initially only have a protos folder with the .proto file as per your requirements



File
Edit
Selection
View
Go
Run
dc_labs
Explorer (Ctrl+Shift+E)
DC_LABS
protos
greet.proto
Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Open a Walkthrough...
Connect to...
Recent
DC D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8
api C:\Users\91985\Documents
EfficientDet_Obj Det D:\PDL\line_in_out_people counter
Refactored_code D:\PDL\line_in_out_people counter\EfficientDet_...
effdet_all D:\PDL\line_in_out_people counter
Eff_MediaPipe_Obj Det_multi poly_current D:\PDL\line_in_out_peo...
Yolo_Obj Det D:\PDL\line_in_out_people counter
patterns C:\Users\91985\Desktop
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs> python -m grpc_tools.protoc -I protos --python_out=. --grpc_python_ou
t=. protos/greet.proto
```

To create my client and server pb files from my proto file, run this cmd:
python -m grpc_tools.protoc -I protos --python_out=. --grpc_python_out=. protos/greet.proto



```
DC          D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8
api         C:\Users\91985\Documents
EfficientDet_Obj Det    D:\PDL\line_in_out_people counter
Refactored_code    D:\PDL\line_in_out_people counter\EfficientDet_...
effdet_all    D:\PDL\line_in_out_people counter
Eff_MediaPipe_Obj Det_multi poly_current    D:\PDL\line_in_out_peo...
Yolo_Obj Det    D:\PDL\line_in_out_people counter
patterns    C:\Users\91985\Desktop
```

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs> python -m grpc_tools.protoc -I protos --python_out=. --grpc_python_ou
t=. protos/greet.proto
C:\Users\91985\AppData\Local\Programs\Python\Python310\lib\site-packages\grpc_tools\protoc.py:21: DeprecationWarning: pkg_resources
is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  import pkg_resources
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs> 
```

Now create the files, server.py and client.py.
Open a split terminal.
Run greet_server.py first, and then greet_client.py next.

Code for server.py:

```
from concurrent import futures
import time
import grpc
import greet_pb2
import greet_pb2_grpc
import sys  # Add sys module

class
GreeterServicer(greet_pb2_grpc.GreeterServic
er):
    def ChattyClientSaysHello(self,
request_iterator, context):
        delayed_reply =
greet_pb2.DelayedReply()
        for request in request_iterator:
            print("ChattyClientSaysHello
Request Made:")
            print(request)

            for countdown in range(5, 0, -1):
                print(f"Time left:
{countdown}")
                time.sleep(1)

            delayed_reply.request.append(req
uest)

        delayed_reply.message = f"You have
sent {len(delayed_reply.request)} messages.
Please expect a delayed response."
        return delayed_reply

    def InteractingHello(self,
request_iterator, context):
        for request in request_iterator:
            print("request from client " +
request.name + " received.", flush=True)
            time.sleep(5)  # Start
processing a new req only after 5 seconds
            print(request, flush=True)

            hello_reply =
greet_pb2.HelloReply()
            hello_reply.message =
f"{request.greeting} {request.name}"

            # print("response to client " +
request.name + " sent", flush=True)
            yield hello_reply
            time.sleep(1)  # Simulate server
processing time

def serve():
    server =
grpc.server(futures.ThreadPoolExecutor(max_w
orkers=10))
    greet_pb2_grpc.add_GreeterServicer_to_se
rver(GreeterServicer(), server)
    server.add_insecure_port("localhost:5005
1")
    server.start()
    server.wait_for_termination()

if __name__ == "__main__":
    serve()
```

Code for client.py:

```
import greet_pb2_grpc
import greet_pb2
import grpc
import asyncio

async def get_client_stream_requests():
    while True:
        name = input("Please enter a name
(or nothing to stop chatting): ")
        print("Request from "+name+ " sent.")

        if name == "":
            break

        hello_request =
greet_pb2.HelloRequest(greeting="Hello",
name=name)
        yield hello_request
        await asyncio.sleep(5)  #to simulate
client waiting for a response from a server
        # print("Response from server to
client "+ name+ " Received.\n---------------
-------------------------------")

async def run_synchronous():
    async with
grpc.aio.insecure_channel('localhost:50051')
as channel:
        stub =
greet_pb2_grpc.GreeterStub(channel)

        delayed_reply = await
stub.ChattyClientSaysHello(get_client_stream
_requests())
        print(f"ChattyClientSaysHello
Response Received: {delayed_reply.message}")
        print("Waiting for server countdown
to end...")

async def run_asynchronous():
    async with
grpc.aio.insecure_channel('localhost:50051')
as channel:
        stub =
greet_pb2_grpc.GreeterStub(channel)
```

```
        async def send_requests():
            async for response in
stub.InteractingHello(get_client_stream_requ
ests()):
                print(f"InteractingHello
Response Received: {response.message}")
                # response for so and so
received
        await asyncio.gather(send_requests(),
send_requests(), send_requests())

def run():
    print("1. Synchronous Communication")
    print("2. Asynchronous Communication")
    communication_type = input("Choose
communication type (1 or 2): ")

    if communication_type == "1":
        asyncio.run(run_synchronous())
    elif communication_type == "2":
        asyncio.run(run_asynchronous())
    else:
        print("Invalid choice. Please choose
1 or 2.")

if __name__ == "__main__":
    run()
```

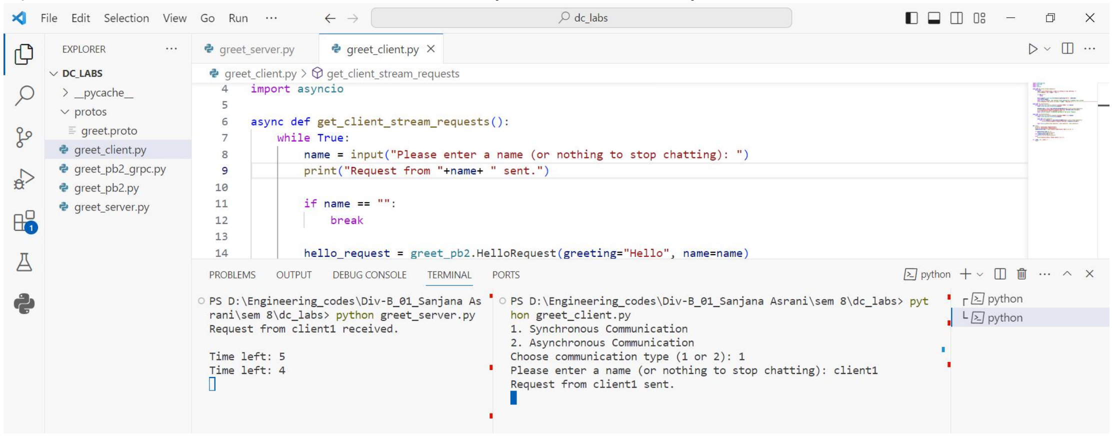
This line: stub = greet_pb2_grpc.GreeterStub(channel)
creates a gRPC client stub, which acts as a client-side proxy for making RPC (Remote Procedure Call) requests to the gRPC server. In gRPC, a stub is a client-side representation of the server's methods. It is generated based on the service definition (in your case, greet_pb2_grpc.GreeterStub), which is defined in the Protocol Buffers file (greet.proto in your case).

The line: await asyncio.gather(send_requests(), send_requests(), send_requests())
is calling the send_requests coroutine three times concurrently. The asyncio.gather function gathers the results of these coroutines into a single result, and it waits until all the coroutines are complete.

1. Synchronous RPC

Output:

1) Client waits until server sends a response to its request

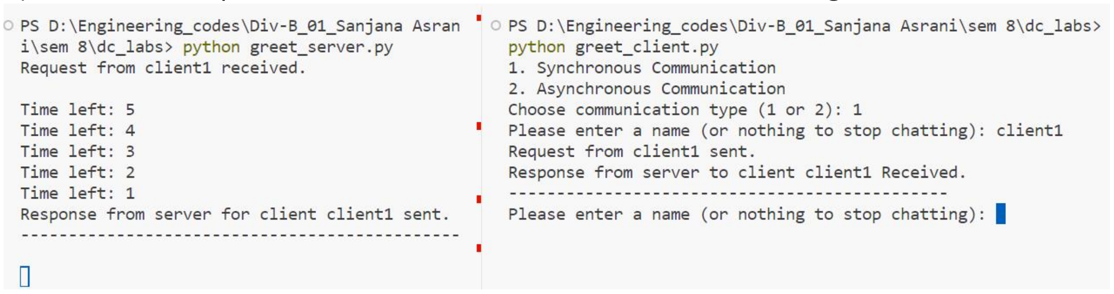


```
import asyncio

async def get_client_stream_requests():
    while True:
        name = input("Please enter a name (or nothing to stop chatting): ")
        print("Request from "+name+ " sent.")

        if name == "":
            break

        hello_request = greet_pb2.HelloRequest(greeting="Hello", name=name)
```

```
PS D:\Engineering_codes\Div-B_01_Sanjana As
rani\sem 8\dc_labs> python greet_server.py
Request from client1 received.

Time left: 5
Time left: 4
```

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs> pyt
hon greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 1
Please enter a name (or nothing to stop chatting): client1
Request from client1 sent.
```

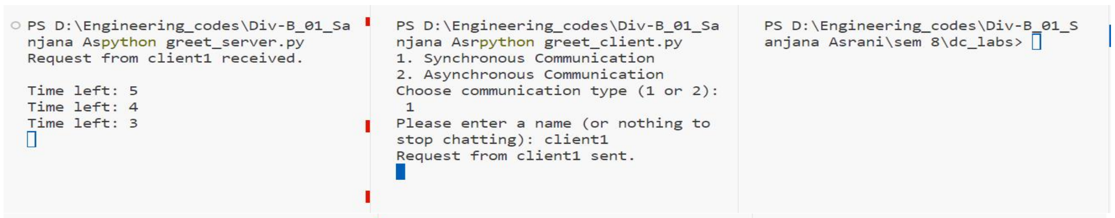
2) As soon as response received, client can send another message:



```
PS D:\Engineering_codes\Div-B_01_Sanjana Asran
i\sem 8\dc_labs> python greet_server.py
Request from client1 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client client1 sent.
----------------------------------------------
```

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs>
python greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 1
Please enter a name (or nothing to stop chatting): client1
Request from client1 sent.
Response from server to client client1 Received.
----------------------------------------------
Please enter a name (or nothing to stop chatting):
```

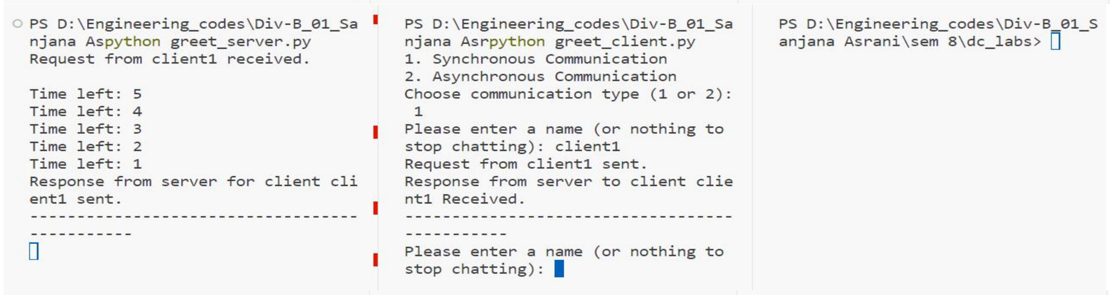
2 clients:



```
PS D:\Engineering_codes\Div-B_01_Sa
njana Aspython greet_server.py
Request from client1 received.

Time left: 5
Time left: 4
Time left: 3
```

```
PS D:\Engineering_codes\Div-B_01_Sa
njana Asrpython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2):
 1
Please enter a name (or nothing to
stop chatting): client1
Request from client1 sent.
```

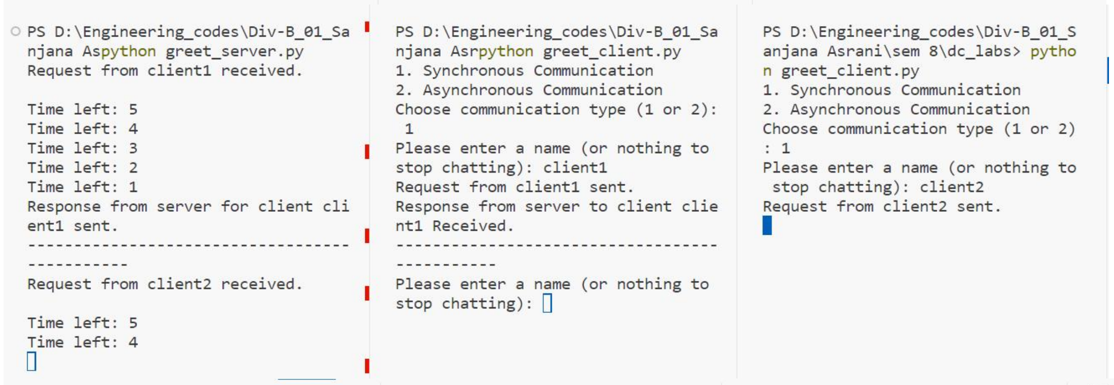
```
PS D:\Engineering_codes\Div-B_01_S
anjana Asrani\sem 8\dc_labs>
```



```
PS D:\Engineering_codes\Div-B_01_Sa
njana Aspython greet_server.py
Request from client1 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client cli
ent1 sent.
-----------------------------------
-----------
```

```
PS D:\Engineering_codes\Div-B_01_Sa
njana Asrpython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2):
 1
Please enter a name (or nothing to
stop chatting): client1
Request from client1 sent.
Response from server to client clie
nt1 Received.
-----------------------------------
-----------
Please enter a name (or nothing to
stop chatting):
```

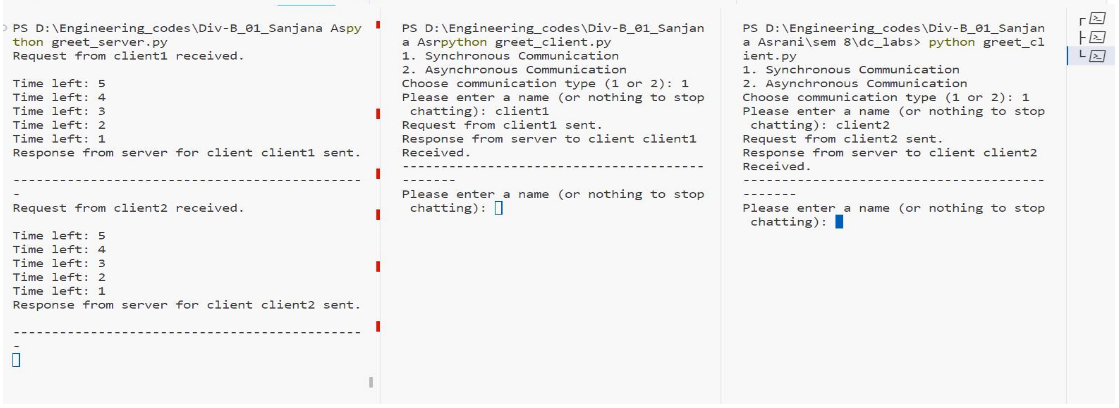
```
PS D:\Engineering_codes\Div-B_01_S
anjana Asrani\sem 8\dc_labs>
```



```
PS D:\Engineering_codes\Div-B_01_Sa
njana Aspython greet_server.py
Request from client1 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client cli
ent1 sent.
-----------------------------------
-----------
Request from client2 received.

Time left: 5
Time left: 4
```

```
PS D:\Engineering_codes\Div-B_01_Sa
njana Asrpython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2):
 1
Please enter a name (or nothing to
stop chatting): client1
Request from client1 sent.
Response from server to client clie
nt1 Received.
-----------------------------------
-----------
Please enter a name (or nothing to
stop chatting):
```

```
PS D:\Engineering_codes\Div-B_01_S
anjana Asrani\sem 8\dc_labs> pytho
n greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2)
: 1
Please enter a name (or nothing to
 stop chatting): client2
Request from client2 sent.
```

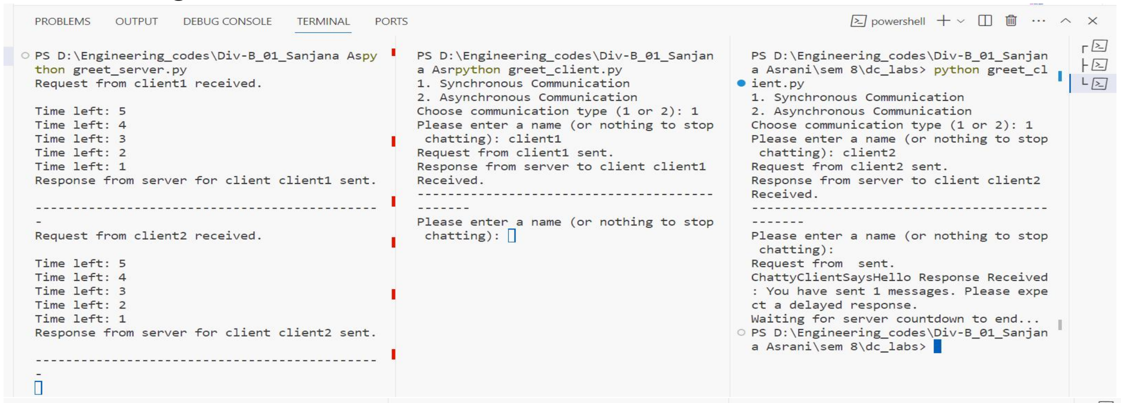


```
PS D:\Engineering_codes\Div-B_01_Sanjana Aspy
thon greet_server.py
Request from client1 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client client1 sent.

---------------------------------------------
-
Request from client2 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client client2 sent.

---------------------------------------------
-
```

```
PS D:\Engineering_codes\Div-B_01_Sanjan
a Asrpython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 1
Please enter a name (or nothing to stop
 chatting): client1
Request from client1 sent.
Response from server to client client1
Received.
---------------------------------------
-------
Please enter a name (or nothing to stop
 chatting):
```

```
PS D:\Engineering_codes\Div-B_01_Sanjan
a Asrani\sem 8\dc_labs> python greet_cl
ient.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 1
Please enter a name (or nothing to stop
 chatting): client2
Request from client2 sent.
Response from server to client client2
Received.
---------------------------------------
-------
Please enter a name (or nothing to stop
 chatting):
```

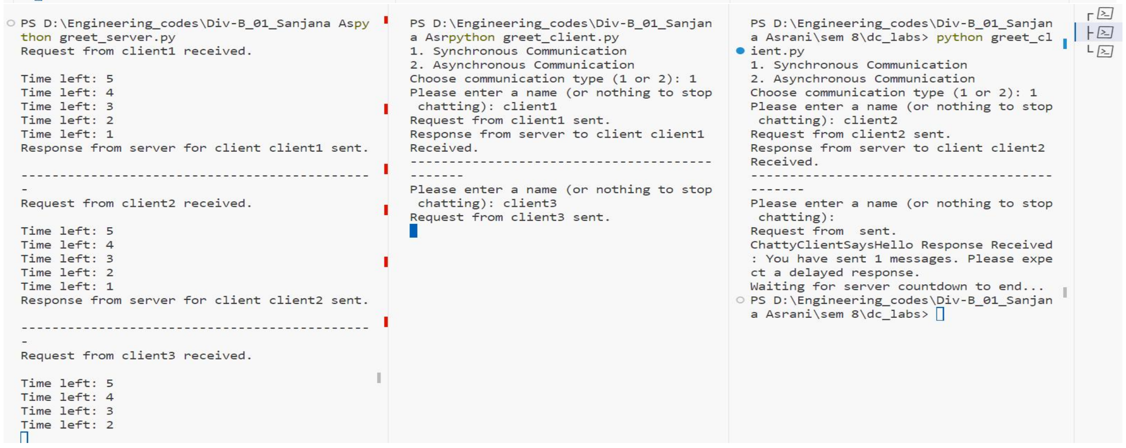
Disconnecting client2:



```
PS D:\Engineering_codes\Div-B_01_Sanjana Aspy
thon greet_server.py
Request from client1 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client client1 sent.

---------------------------------------------
-
Request from client2 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client client2 sent.

---------------------------------------------
-
```

```
PS D:\Engineering_codes\Div-B_01_Sanjan
a Asrpython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 1
Please enter a name (or nothing to stop
 chatting): client1
Request from client1 sent.
Response from server to client client1
Received.
---------------------------------------
-------
Please enter a name (or nothing to stop
 chatting):
```

```
PS D:\Engineering_codes\Div-B_01_Sanjan
a Asrani\sem 8\dc_labs> python greet_cl
ient.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 1
Please enter a name (or nothing to stop
 chatting): client2
Request from client2 sent.
Response from server to client client2
Received.
---------------------------------------
-------
Please enter a name (or nothing to stop
 chatting):
Request from  sent.
ChattyClientSaysHello Response Received
: You have sent 1 messages. Please expe
ct a delayed response.
Waiting for server countdown to end...
PS D:\Engineering_codes\Div-B_01_Sanjan
a Asrani\sem 8\dc_labs>
```

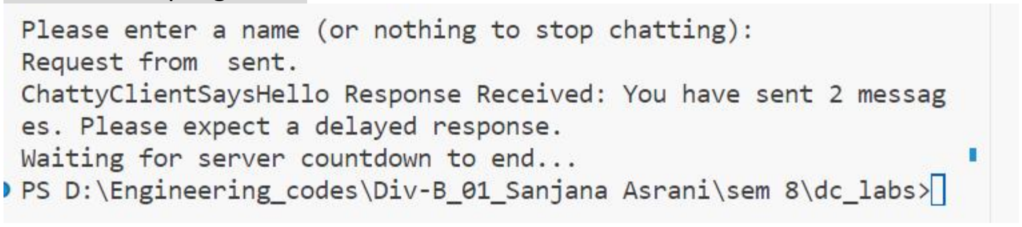


```
PS D:\Engineering_codes\Div-B_01_Sanjana Aspy
thon greet_server.py
Request from client1 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client client1 sent.

---------------------------------------------
-
Request from client2 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client client2 sent.

---------------------------------------------
-
Request from client3 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
```

```
PS D:\Engineering_codes\Div-B_01_Sanjan
a Asrpython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 1
Please enter a name (or nothing to stop
 chatting): client1
Request from client1 sent.
Response from server to client client1
Received.
---------------------------------------
-------
Please enter a name (or nothing to stop
 chatting): client3
Request from client3 sent.
```

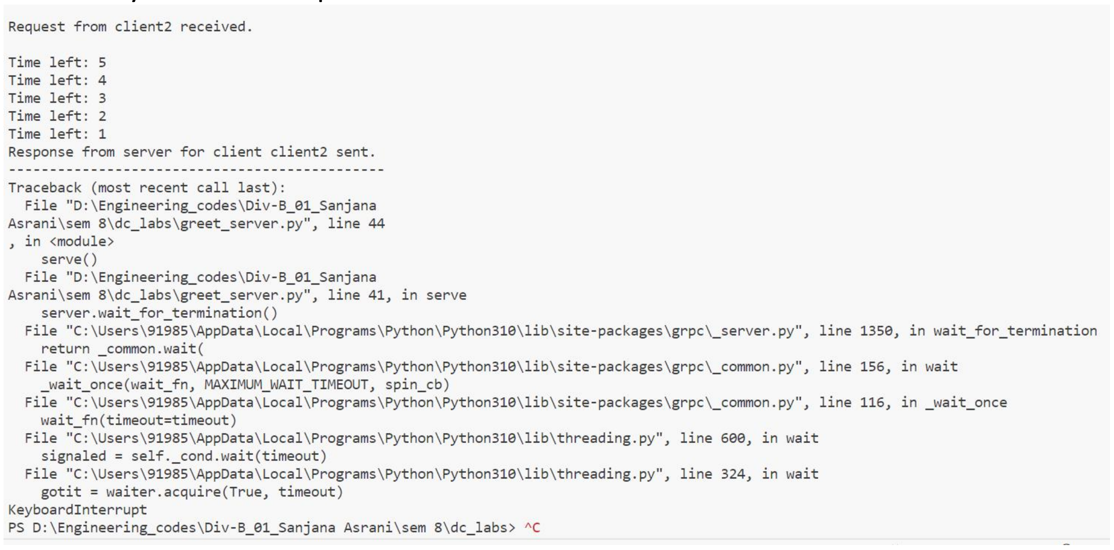
```
PS D:\Engineering_codes\Div-B_01_Sanjan
a Asrani\sem 8\dc_labs> python greet_cl
ient.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 1
Please enter a name (or nothing to stop
 chatting): client2
Request from client2 sent.
Response from server to client client2
Received.
---------------------------------------
-------
Please enter a name (or nothing to stop
 chatting):
Request from  sent.
ChattyClientSaysHello Response Received
: You have sent 1 messages. Please expe
ct a delayed response.
Waiting for server countdown to end...
PS D:\Engineering_codes\Div-B_01_Sanjan
a Asrani\sem 8\dc_labs>
```

To close the programs,



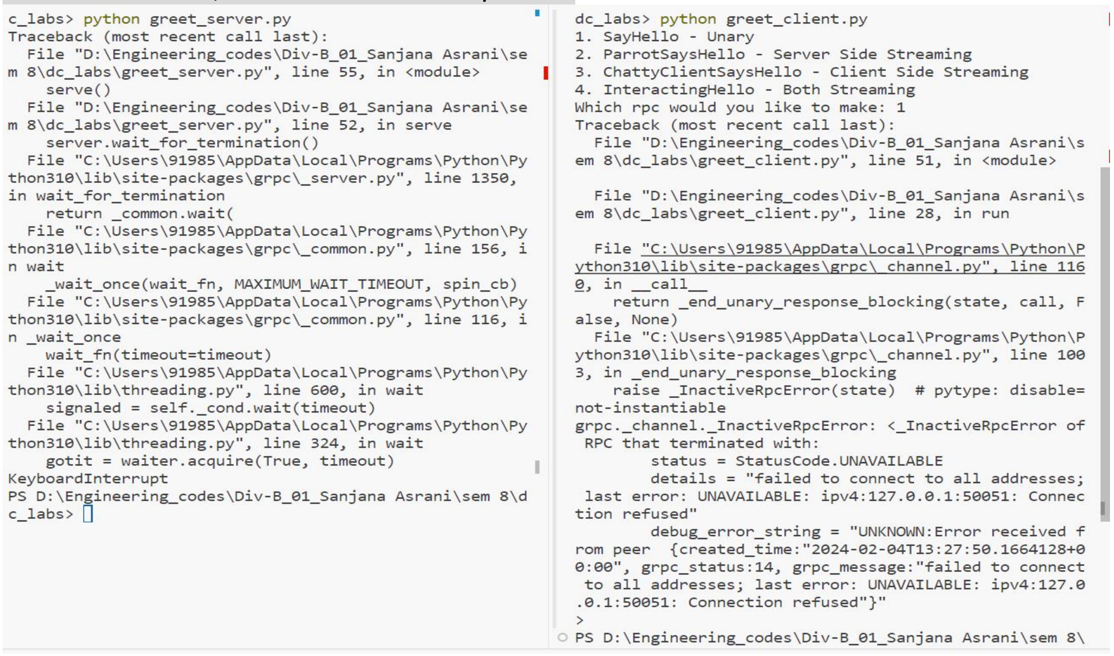
```
Please enter a name (or nothing to stop chatting):
Request from  sent.
ChattyClientSaysHello Response Received: You have sent 2 messag
es. Please expect a delayed response.
Waiting for server countdown to end...
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs>
```

Send nothing from client to server, client closes down.
Send a keyboard interrupt on serverside.



```
Request from client2 received.

Time left: 5
Time left: 4
Time left: 3
Time left: 2
Time left: 1
Response from server for client client2 sent.
----------------------------------------------
Traceback (most recent call last):
  File "D:\Engineering_codes\Div-B_01_Sanjana
Asrani\sem 8\dc_labs\greet_server.py", line 44
, in <module>
    serve()
  File "D:\Engineering_codes\Div-B_01_Sanjana
Asrani\sem 8\dc_labs\greet_server.py", line 41, in serve
    server.wait_for_termination()
  File "C:\Users\91985\AppData\Local\Programs\Python\Python310\lib\site-packages\grpc\_server.py", line 1350, in wait_for_termination
    return _common.wait(
  File "C:\Users\91985\AppData\Local\Programs\Python\Python310\lib\site-packages\grpc\_common.py", line 156, in wait
    _wait_once(wait_fn, MAXIMUM_WAIT_TIMEOUT, spin_cb)
  File "C:\Users\91985\AppData\Local\Programs\Python\Python310\lib\site-packages\grpc\_common.py", line 116, in _wait_once
    wait_fn(timeout=timeout)
  File "C:\Users\91985\AppData\Local\Programs\Python\Python310\lib\threading.py", line 600, in wait
    signaled = self._cond.wait(timeout)
  File "C:\Users\91985\AppData\Local\Programs\Python\Python310\lib\threading.py", line 324, in wait
    gotit = waiter.acquire(True, timeout)
KeyboardInterrupt
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs> ^C
```

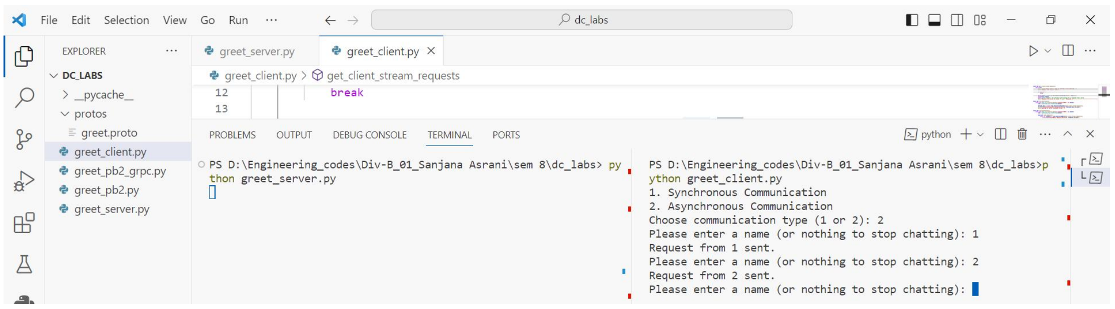
If server crashes, and client tries to operate:



```
c_labs> python greet_server.py
Traceback (most recent call last):
  File "D:\Engineering_codes\Div-B_01_Sanjana Asrani\se
m 8\dc_labs\greet_server.py", line 55, in <module>
    serve()
  File "D:\Engineering_codes\Div-B_01_Sanjana Asrani\se
m 8\dc_labs\greet_server.py", line 52, in serve
    server.wait_for_termination()
  File "C:\Users\91985\AppData\Local\Programs\Python\Py
thon310\lib\site-packages\grpc\_server.py", line 1350,
in wait_for_termination
    return _common.wait(
  File "C:\Users\91985\AppData\Local\Programs\Python\Py
thon310\lib\site-packages\grpc\_common.py", line 156, i
n wait
    _wait_once(wait_fn, MAXIMUM_WAIT_TIMEOUT, spin_cb)
  File "C:\Users\91985\AppData\Local\Programs\Python\Py
thon310\lib\site-packages\grpc\_common.py", line 116, i
n _wait_once
    wait_fn(timeout=timeout)
  File "C:\Users\91985\AppData\Local\Programs\Python\Py
thon310\lib\threading.py", line 600, in wait
    signaled = self._cond.wait(timeout)
  File "C:\Users\91985\AppData\Local\Programs\Python\Py
thon310\lib\threading.py", line 324, in wait
    gotit = waiter.acquire(True, timeout)
KeyboardInterrupt
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\d
c_labs>
```

```
dc_labs> python greet_client.py
1. SayHello - Unary
2. ParrotSaysHello - Server Side Streaming
3. ChattyClientSaysHello - Client Side Streaming
4. InteractingHello - Both Streaming
Which rpc would you like to make: 1
Traceback (most recent call last):
  File "D:\Engineering_codes\Div-B_01_Sanjana Asrani\s
em 8\dc_labs\greet_client.py", line 51, in <module>

  File "D:\Engineering_codes\Div-B_01_Sanjana Asrani\s
em 8\dc_labs\greet_client.py", line 28, in run

  File "C:\Users\91985\AppData\Local\Programs\Python\P
ython310\lib\site-packages\grpc\_channel.py", line 116
0, in __call__
    return _end_unary_response_blocking(state, call, F
alse, None)
  File "C:\Users\91985\AppData\Local\Programs\Python\P
ython310\lib\site-packages\grpc\_channel.py", line 100
3, in _end_unary_response_blocking
    raise _InactiveRpcError(state)  # pytype: disable=
not-instantiable
grpc._channel._InactiveRpcError: <_InactiveRpcError of
 RPC that terminated with:
        status = StatusCode.UNAVAILABLE
        details = "failed to connect to all addresses;
 last error: UNAVAILABLE: ipv4:127.0.0.1:50051: Connec
tion refused"
        debug_error_string = "UNKNOWN:Error received f
rom peer  {created_time:"2024-02-04T13:27:50.1664128+0
0:00", grpc_status:14, grpc_message:"failed to connect
 to all addresses; last error: UNAVAILABLE: ipv4:127.0
.0.1:50051: Connection refused"}"
>
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\
```

2. Asynchronous communication



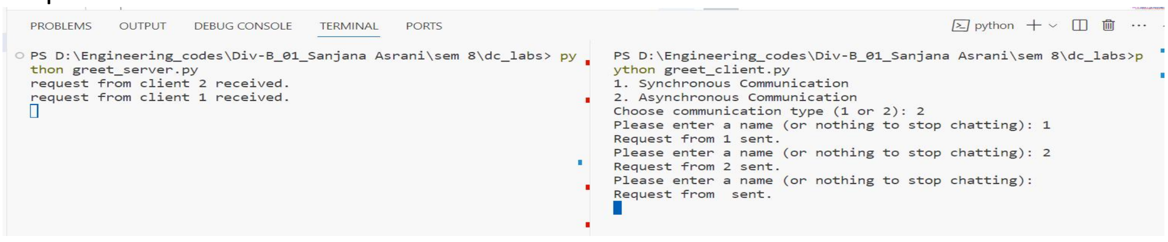
```
greet_client.py > get_client_stream_requests
                break
```

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs> py
thon greet_server.py
```

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs>p
ython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 2
Please enter a name (or nothing to stop chatting): 1
Request from 1 sent.
Please enter a name (or nothing to stop chatting): 2
Request from 2 sent.
Please enter a name (or nothing to stop chatting):
```

Neither 1, nor 2 has received a response from server, but I still can make requests from client to server.

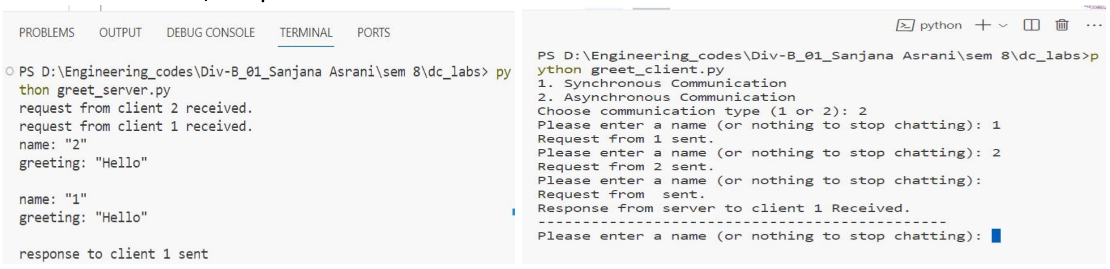
Requets received:



```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs> py
thon greet_server.py
request from client 2 received.
request from client 1 received.
```

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs>p
ython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 2
Please enter a name (or nothing to stop chatting): 1
Request from 1 sent.
Please enter a name (or nothing to stop chatting): 2
Request from 2 sent.
Please enter a name (or nothing to stop chatting):
Request from  sent.
```

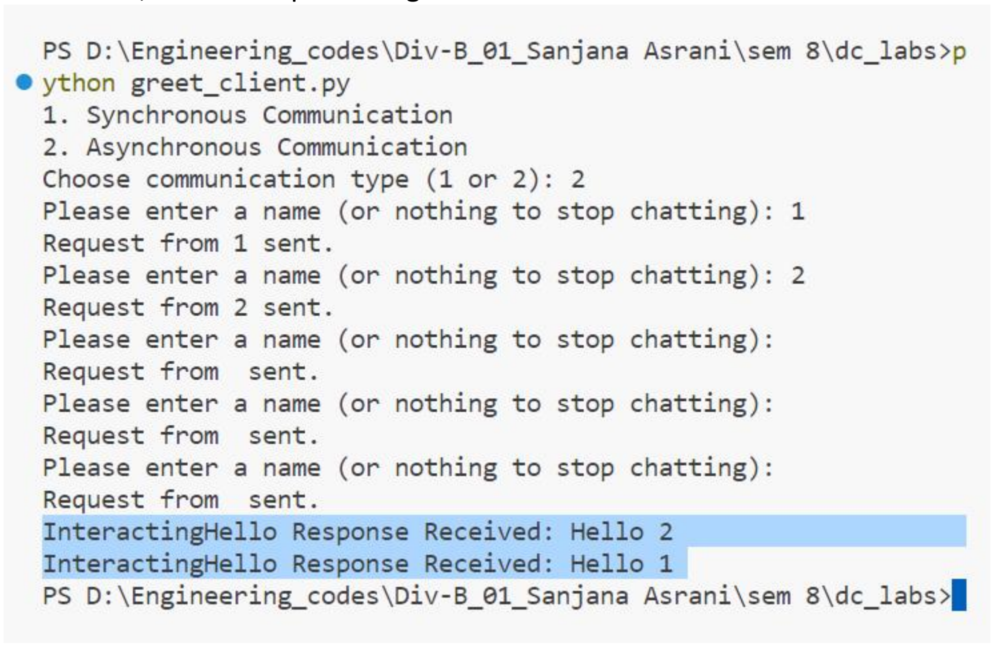
After 5 seconds, response sent for client1:



```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs> py
thon greet_server.py
request from client 2 received.
request from client 1 received.
name: "2"
greeting: "Hello"

name: "1"
greeting: "Hello"

response to client 1 sent
```

```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs>p
ython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 2
Please enter a name (or nothing to stop chatting): 1
Request from 1 sent.
Please enter a name (or nothing to stop chatting): 2
Request from 2 sent.
Please enter a name (or nothing to stop chatting):
Request from  sent.
Response from server to client 1 Received.
----------------------------------------------------
Please enter a name (or nothing to stop chatting):
```

At the end, when I stop chatting for 3 times:



```
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs>p
ython greet_client.py
1. Synchronous Communication
2. Asynchronous Communication
Choose communication type (1 or 2): 2
Please enter a name (or nothing to stop chatting): 1
Request from 1 sent.
Please enter a name (or nothing to stop chatting): 2
Request from 2 sent.
Please enter a name (or nothing to stop chatting):
Request from  sent.
Please enter a name (or nothing to stop chatting):
Request from  sent.
Please enter a name (or nothing to stop chatting):
Request from  sent.
InteractingHello Response Received: Hello 2
InteractingHello Response Received: Hello 1
PS D:\Engineering_codes\Div-B_01_Sanjana Asrani\sem 8\dc_labs>
```

The responses are received.