DS Lab6
Rhea Adhikari
190905156
Roll No : 23

1. The Manipal Foodie is a renowned automated food processing outlet known for its tiffin service to students. The various processes involved are food production, filling and packing. Every day more than 3000 orders are received on an average from the students in manipal. There are total of 4 production lines for orders received from KMC, MIT, TAPMI and SOLS students, each of them has a digital clock which needs to be in synchronization with the master clock. The master clock mounted in the testing lab controls the entire clock system. Design an appropriate solution using Berkeley's algorithm for the above scenario. Assume that the clocks at the institutes are slave/clients.

q1_main.py:

```
from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time
client_data = {}
def startRecieveingClockTime(connector, address):
while True:
clock_time_string = connector.recv(1024).decode()
clock_time = parser.parse(clock_time_string)
clock_time_diff = datetime.datetime.now() - clock_time
client_data[address] = {
"clock_time" : clock_time,
"time_difference" : clock_time_diff,
"connector" : connector
}
print("Client Data updated with: "+ str(address), end = "\n\n")
time.sleep(5)
def startConnecting(master_server):
while True:
master_slave_connector, addr = master_server.accept()
slave_address = str(addr[0]) + ":" + str(addr[1])
print(slave_address + " got connected successfully")
current_thread = threading.Thread(
target = startRecieveingClockTime,
args = (master_slave_connector, slave_address, ))
current_thread.start()
def getAverageClockDiff():
current_client_data = client_data.copy()
time_difference_list = list(client['time_difference'] for client_addr, client in
client_data.items())
sum_of_clock_difference = sum(time_difference_list, datetime.timedelta(0, 0))
average_clock_difference = sum_of_clock_difference / len(client_data)return
average_clock_difference
def synchronizeAllClocks():
while True:
```

```
print("New synchroniztion cycle started.")
print("Number of clients to be synchronized: " + str(len(client_data)))
if len(client_data) > 0:
average_clock_difference = getAverageClockDiff()
for client_addr, client in client_data.items():
try:
synchronized_time = datetime.datetime.now() +
average_clock_difference
client['connector'].send(str(synchronized_time).encode())
except Exception as e:
print("Something went wrong while sending synchronized
time through " + str(client_addr))
else :
print("No client data. Synchronization not applicable.")
print("\n\n")
time.sleep(5)
def initiateClockServer(port = 8080):
master_server = socket.socket()
master_server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
print("The Manipal Foodie\n")
master_server.bind(('', port))
master_server.listen(10)
print("Clock server print\n")
print("Connecitng to production lines...\n")
master_thread = threading.Thread(
target = startConnecting,
args = (master_server, ))
master_thread.start()
print("Starting synchronization parallely...\n")
sync_thread = threading.Thread(
target = synchronizeAllClocks,
args = ())
sync_thread.start()
if __name__ == '__main__':
initiateClockServer(port = 8080)
```

mit.py:

```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time
def startSendingTime(slave_client):
while True:slave_client.send(str(datetime.datetime.now()).encode())
print("MIT time sent successfully", end = "\n\n")
time.sleep(5)
def startReceivingTime(slave_client):
while True:
Synchronized_time = parser.parse(slave_client.recv(1024).decode())
print("Synchronized time at the client is: " + str(Synchronized_time), end = "\n\n")
def initiateSlaveClient(port = 8080):
```

```
slave_client = socket.socket()
slave_client.connect(('127.0.0.1', port))
print("Starting to receive time from server\n")
send_time_thread = threading.Thread(
target = startSendingTime,
args = (slave_client, ))
send_time_thread.start()
print("Starting to recieving synchronized time from server\n")
receive_time_thread = threading.Thread(
target = startReceivingTime,
args = (slave_client, ))
receive_time_thread.start()
if __name__ == '__main__':
initiateSlaveClient(port = 8080)
```

kmc.py:

```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time
def startSendingTime(slave_client):
while True:
slave_client.send(str(datetime.datetime.now()).encode())
print("KMC time sent successfully", end = "\n\n")
time.sleep(5)
def startReceivingTime(slave_client):
while True:
Synchronized_time = parser.parse(slave_client.recv(1024).decode())
print("Synchronized time at the client is: " + str(Synchronized_time), end = "\n\n")
def initiateSlaveClient(port = 8080):
slave_client = socket.socket()
slave_client.connect(('127.0.0.1', port))
print("Starting to receive time from server\n")
send_time_thread = threading.Thread(
target = startSendingTime,args = (slave_client, ))
send_time_thread.start()
print("Starting to recieving synchronized time from server\n")
receive_time_thread = threading.Thread(
target = startReceivingTime,
args = (slave_client, ))
receive_time_thread.start()
if __name__ == '__main__':
initiateSlaveClient(port = 8080)
```

tapmi.py:

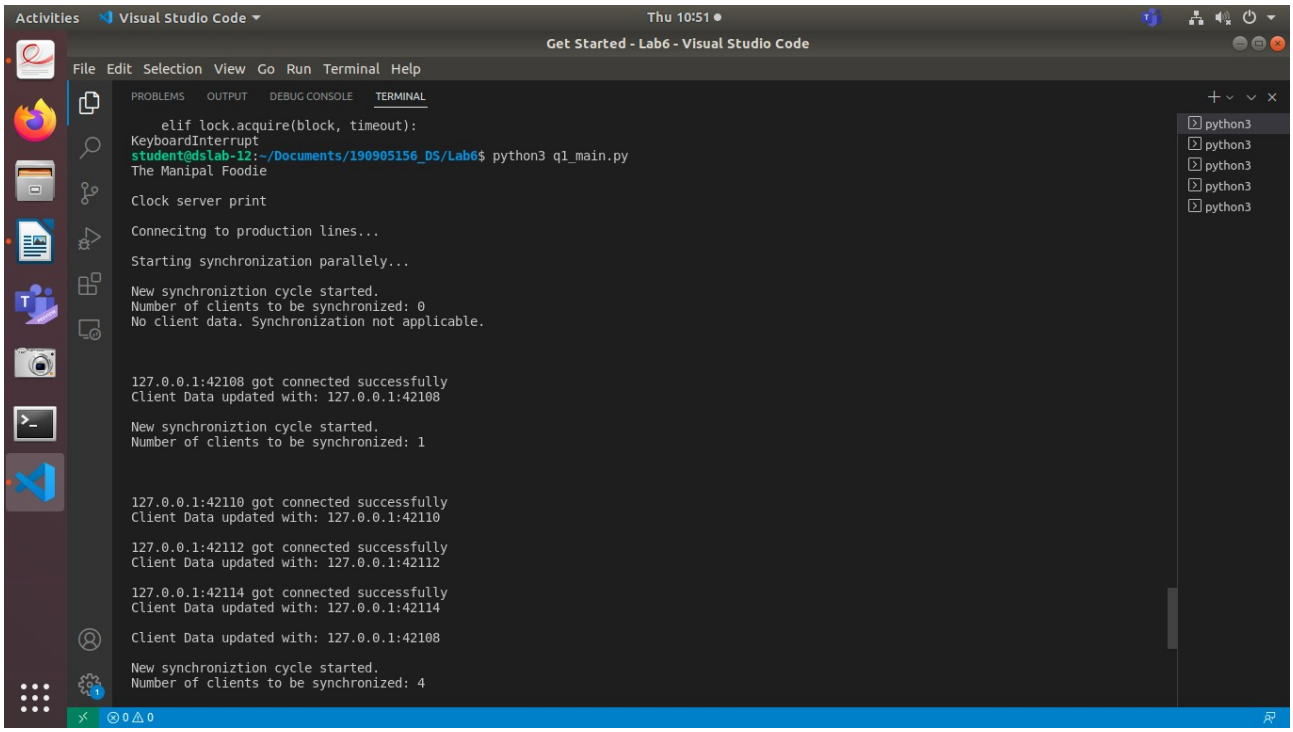
```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
```

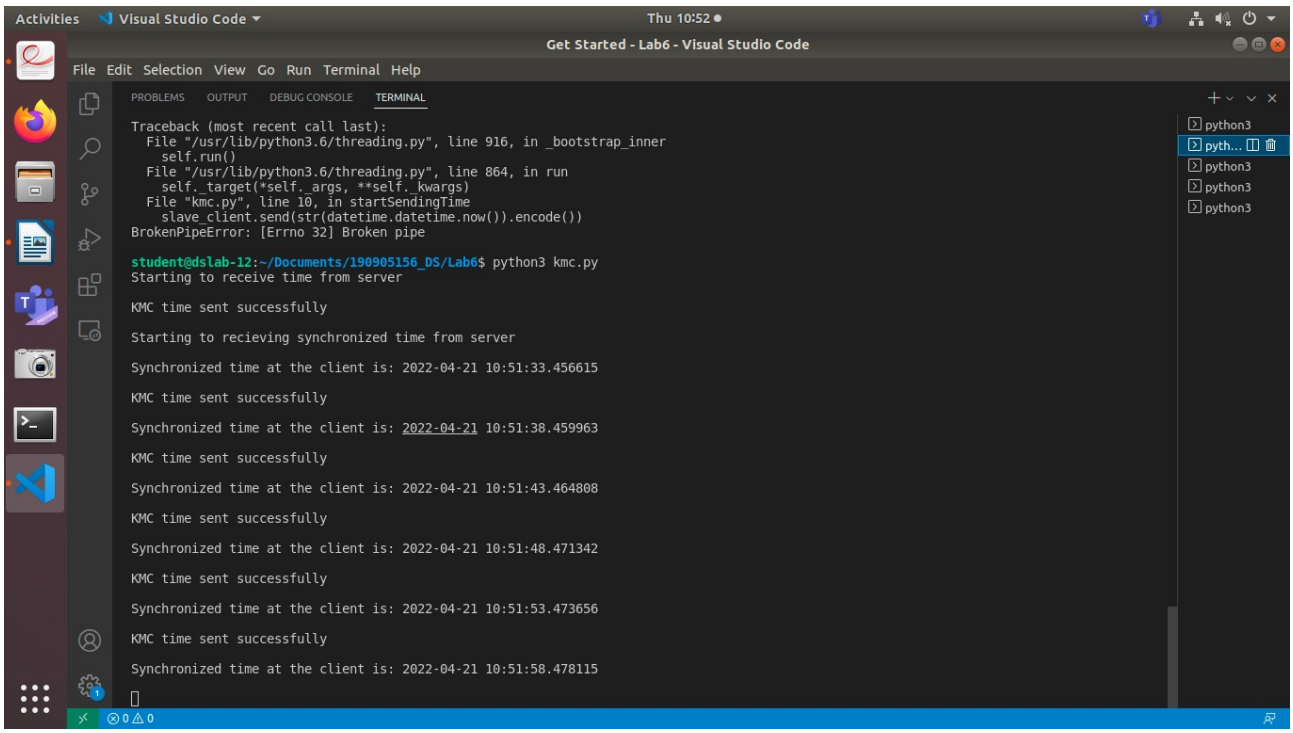
```
import time
def startSendingTime(slave_client):
while True:
slave_client.send(str(datetime.datetime.now()).encode())
print("TAPMI time sent successfully", end = "\n\n")
time.sleep(5)
def startReceivingTime(slave_client):
while True:
Synchronized_time = parser.parse(slave_client.recv(1024).decode())
print("Synchronized time at the client is: " + str(Synchronized_time), end = "\n\n")
def initiateSlaveClient(port = 8080):
slave_client = socket.socket()
slave_client.connect(('127.0.0.1', port))
print("Starting to receive time from server\n")
send_time_thread = threading.Thread(
target = startSendingTime,
args = (slave_client, ))
send_time_thread.start()
print("Starting to recieving synchronized time from server\n")
receive_time_thread = threading.Thread(
target = startReceivingTime,
args = (slave_client, ))
receive_time_thread.start()
if __name__ == '__main__':
initiateSlaveClient(port = 8080)
```

sols.py:

```
from timeit import default_timer as timer
from dateutil import parser
import threadingimport datetime
import socket
import time
def startSendingTime(slave_client):
while True:
slave_client.send(str(datetime.datetime.now()).encode())
print("SOLS time sent successfully", end = "\n\n")
time.sleep(5)
def startReceivingTime(slave_client):
while True:
Synchronized_time = parser.parse(slave_client.recv(1024).decode())
print("Synchronized time at the client is: " + str(Synchronized_time), end = "\n\n")
def initiateSlaveClient(port = 8080):
slave_client = socket.socket()
slave_client.connect(('127.0.0.1', port))
print("Starting to receive time from server\n")
send_time_thread = threading.Thread(
target = startSendingTime,
args = (slave_client, ))
send_time_thread.start()
print("Starting to recieving synchronized time from server\n")
receive_time_thread = threading.Thread(
target = startReceivingTime,
```

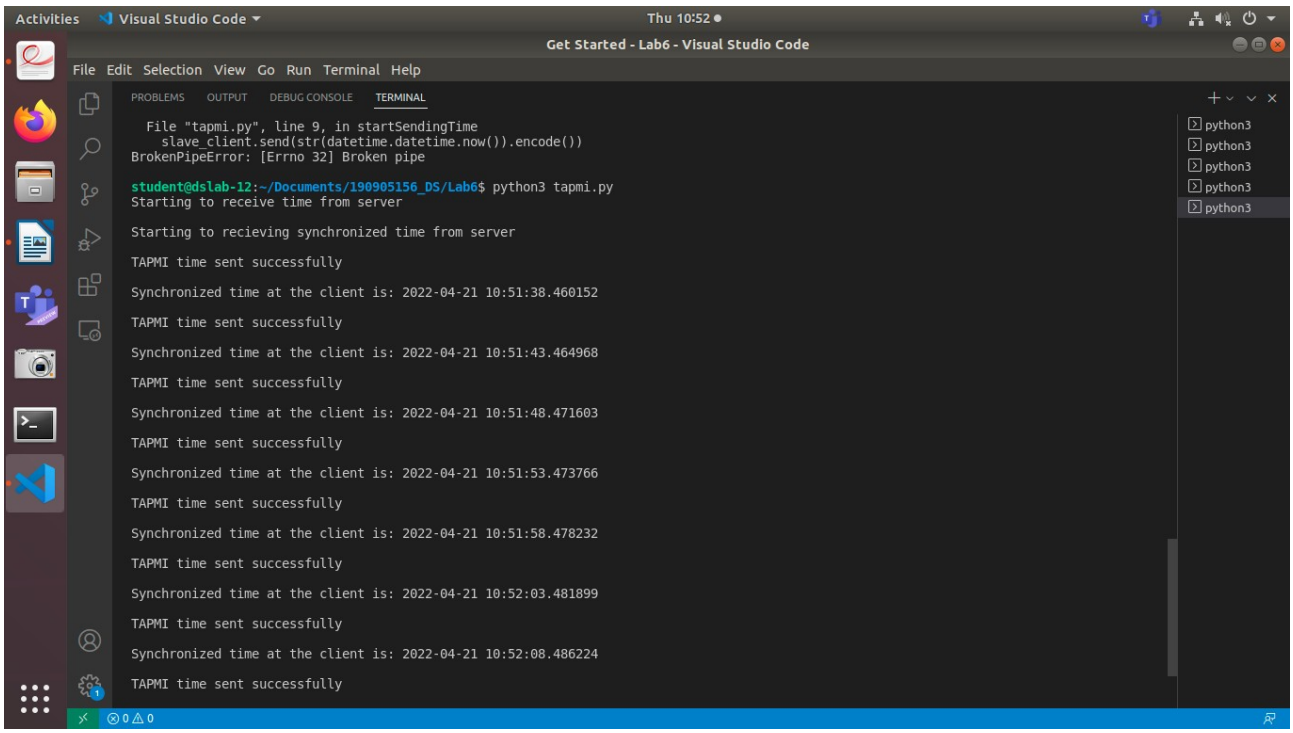
```
args = (slave_client, ))
receive_time_thread.start()
if __name__ == '__main__':
initiateSlaveClient(port = 8080)
```





```
    self.run()
  File "/usr/lib/python3.6/threading.py", line 864, in run
    self._target(*self._args, **self._kwargs)
  File "mit.py", line 9, in startSendingTime
    slave_client.send(str(datetime.datetime.now()).encode())
BrokenPipeError: [Errno 32] Broken pipe

student@dslab-12:~/Documents/190905156_DS/Lab6$ python3 mit.py
Starting to receive time from server

Starting to recieving synchronized time from server

MIT time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:38.460073

MIT time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:43.464904

MIT time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:48.471496

MIT time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:53.473726

MIT time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:58.478195

MIT time sent successfully

Synchronized time at the client is: 2022-04-21 10:52:03.481850

MIT time sent successfully
```

```
student@dslab-12:~/Documents/190905156_DS/Lab6$ python3 sols.py
Starting to receive time from server

SOLS time sent successfully

Starting to recieving synchronized time from server

Synchronized time at the client is: 2022-04-21 10:51:38.460114

SOLS time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:43.464939

SOLS time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:48.471558

SOLS time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:53.473747

SOLS time sent successfully

Synchronized time at the client is: 2022-04-21 10:51:58.478216

SOLS time sent successfully

Synchronized time at the client is: 2022-04-21 10:52:03.481877

SOLS time sent successfully

Synchronized time at the client is: 2022-04-21 10:52:08.486210

SOLS time sent successfully

Synchronized time at the client is: 2022-04-21 10:52:13.489655
```



2. Manipal Buddy is a banking and education application for the students and staff of MIT, Manipal. Mr Vinay, a sixth semester student wants to pay the end semester exams fees for a re-registered course. He simultaneously wishes to register for a course on NPTEL through the app. To register for exam he uses the mobile app whereas to register for NPTEL course he uses his laptop to log in. As he needs to finish both the registrations on the same day, he tries to do both the tasks simultaneously. Analyse and demonstrate using a program how Cristian's algorithm can be used in the above case to synchronize the clocks. Assume the relevant parameters.

q2_server.py:

```
import socket
import datetime
import time
def initiateClockServer():s = socket.socket()
print("Manipal Buddy Banking")
port = 8011
s.bind(('', port))
s.listen(5)
print("Waiting for client...")
while True:
connection, address = s.accept()
print('Server connected to', address)
connection.send(str(datetime.datetime.now()).encode())
connection.close()
if __name__ == '__main__':
initiateClockServer()
```

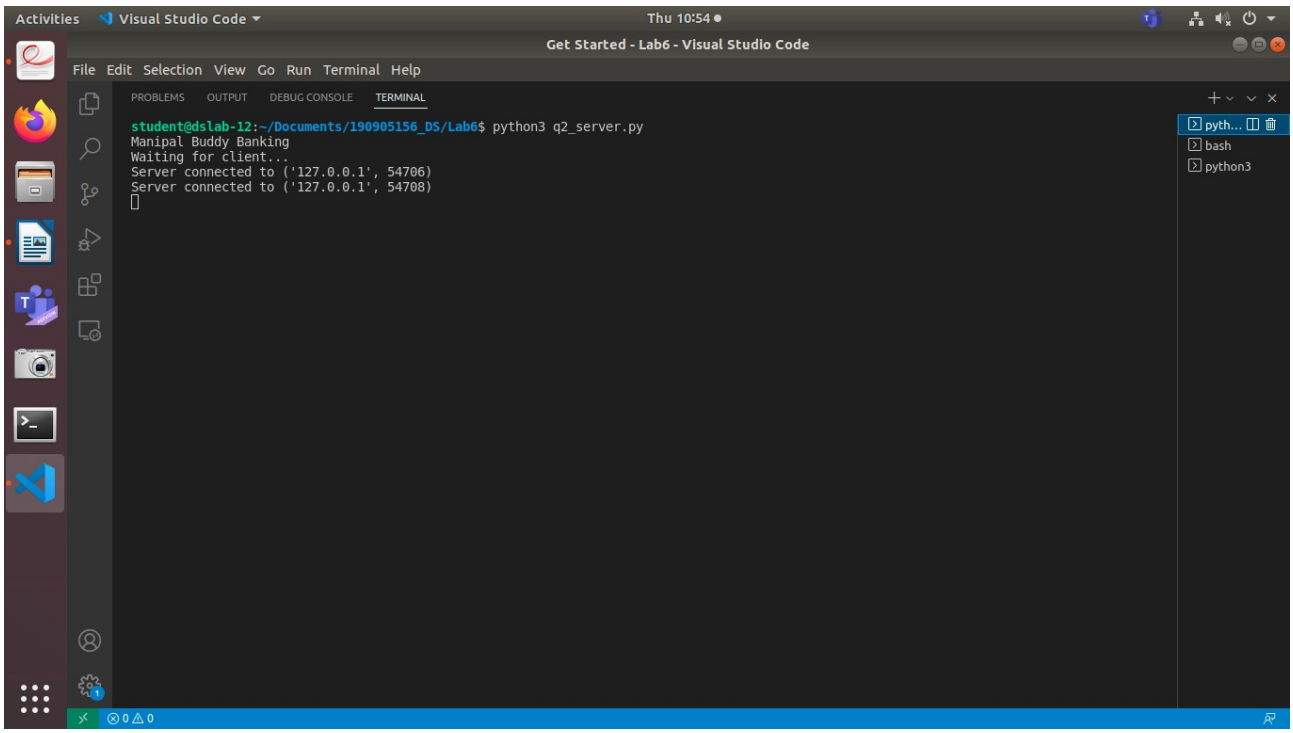
mobile_app.py:

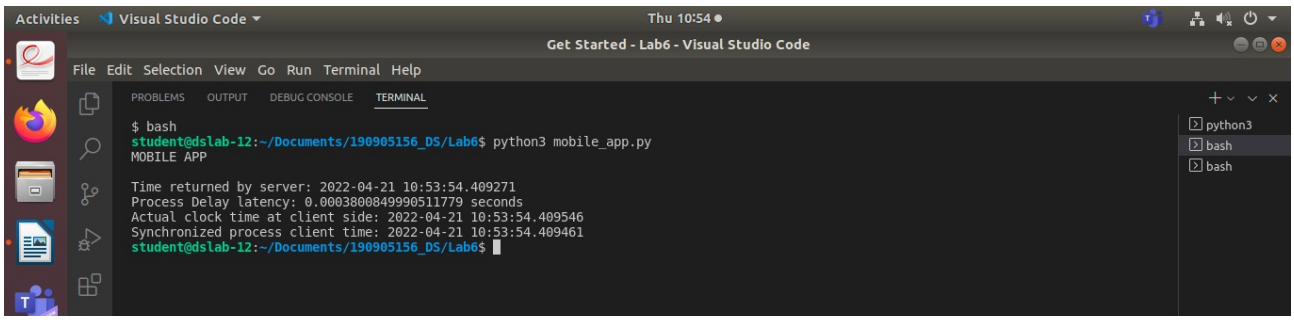
```
import socket
import datetime
import time
```

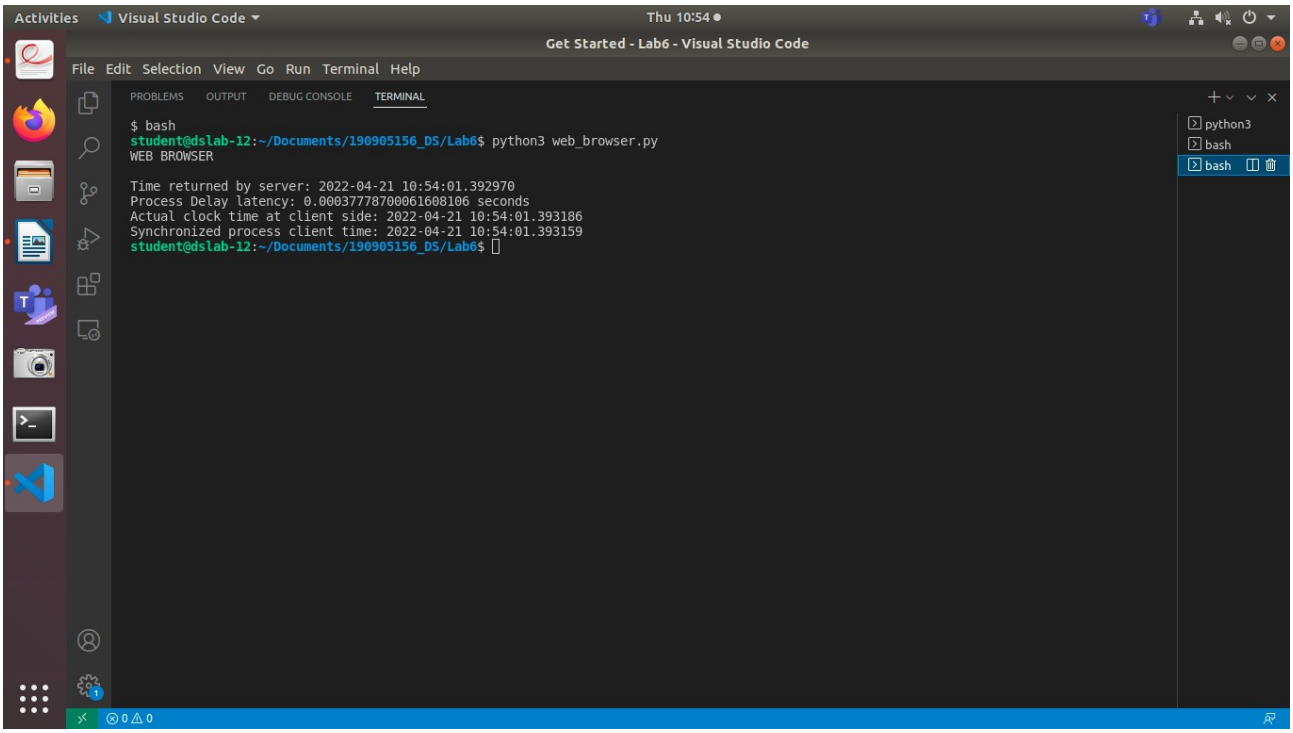
```
from dateutil import parser
from timeit import default_timer as timer
def synchronizeTime():
print("MOBILE APP\n")
s = socket.socket()
port = 8011
s.connect(('127.0.0.1', port))
request_time = timer()
server_time = parser.parse(s.recv(1024).decode())
response_time = timer()
actual_time = datetime.datetime.now()
print("Time returned by server: " + str(server_time))
process_delay_latency = response_time - request_time
print("Process Delay latency: " + str(process_delay_latency) + " seconds")
print("Actual clock time at client side: " + str(actual_time))
client_time = server_time + datetime.timedelta(seconds = (process_delay_latency) / 2)
print("Synchronized process client time: " + str(client_time))
time.sleep(10)
s.close()
if __name__ == '__main__':
synchronizeTime()
```

web_browser.py:

```
import socket
import datetime
import time
from dateutil import parser
from timeit import default_timer as timer
def synchronizeTime():
print("WEB BROWSER\n")
s = socket.socket()port = 8011
s.connect(('127.0.0.1', port))
request_time = timer()
server_time = parser.parse(s.recv(1024).decode())
response_time = timer()
actual_time = datetime.datetime.now()
print("Time returned by server: " + str(server_time))
process_delay_latency = response_time - request_time
print("Process Delay latency: " + str(process_delay_latency) + " seconds")
print("Actual clock time at client side: " + str(actual_time))
client_time = server_time + datetime.timedelta(seconds = (process_delay_latency) / 2)
print("Synchronized process client time: " + str(client_time))
time.sleep(10)
s.close()
if __name__ == '__main__':
synchronizeTime()
```


Activities
Visual Studio Code
Thu 10:54
Get Started - Lab6 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
student@dslab-12:~/Documents/190905156_DS/Lab6$ python3 q2_server.py
Manipal Buddy Banking
Waiting for client...
Server connected to ('127.0.0.1', 54706)
Server connected to ('127.0.0.1', 54708)
pyth...
bash
python3


Activities
Visual Studio Code
Thu 10:54
Get Started - Lab6 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
$ bash
student@dslab-12:~/Documents/190905156_DS/Lab6$ python3 mobile_app.py
MOBILE APP
Time returned by server: 2022-04-21 10:53:54.409271
Process Delay latency: 0.0003800849990511779 seconds
Actual clock time at client side: 2022-04-21 10:53:54.409546
Synchronized process client time: 2022-04-21 10:53:54.409461
student@dslab-12:~/Documents/190905156_DS/Lab6$
python3
bash
bash


Activities
Visual Studio Code
Thu 10:54
Get Started - Lab6 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
$ bash
student@dslab-12:~/Documents/190905156_DS/Lab6$ python3 web_browser.py
WEB BROWSER
Time returned by server: 2022-04-21 10:54:01.392970
Process Delay latency: 0.00037778700061608106 seconds
Actual clock time at client side: 2022-04-21 10:54:01.393186
Synchronized process client time: 2022-04-21 10:54:01.393159
student@dslab-12:~/Documents/190905156_DS/Lab6$
python3
bash
bash