Digital Payment Platform

RollNo	Name	Contribution in case study
CB.EN.U4CSE19126	Darshan M	Software Team
CB.EN.U4CSE19128	Mahima Anjali Lolla	Finance Team
CB.EN.U4CSE19130	Mathan Karthick M P	Operations Team

Description of the case study:

The aim of this case study is to analyze the different types of networks used in a digital payment platform such as Gpay, PhonePe etc. This case study would help us in understanding different types of network infrastructure used in real-world scenarios. For this case study we have assumed 3 teams, which are : Software Team, Finance Team and Operations Team

Software Team: Works on the continuous development and deployment of the mobile app. It is the most essential team of our case study and it requires a large number of computers since the developers use it. So, let's assume they use 20 computers (Mahima)

Finance Team: This team deals with auditing all the transactions and maintaining a strong and close relationship with banks and businesses that our app deals with. It requires close to 10 computers on the network. (Darshan)

Operations Team: This team deals with customer complaints. It also helps in the day to day requirements of the firm. This team requires 10 computers on the network. (Mathan)

Operations Team:

Client	Windows, MacOS, Linux, Android, iOS
Server	**IBM Cloud Hosting** Configuration: Intel Xeon E3-1270 v6 4 Cores, 3.80 GHz 16 GB RAM 1 x 1 TB HDD CentOS 20 TB bandwidth*

List of Network performance parameters:

Parameter	Meaning	Formula
Bandwidth	Bandwidth is the capacity of a wired or wireless network communications link to transmit the maximum amount of data from one point to another over a computer network or internet connection in a given amount of time	Expressed as bits per second (bps), modern network links have greater capacity, which is typically measured in millions of bits per second (megabits per second, or Mbps) or billions of bits per second (gigabits per second, or Gbps).
Throughput	Throughput measures the percentage of data packets that are successfully being sent; a low throughput means there are a lot of failed or dropped packets that need to be sent again.	
Packet Loss	Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Due to network congestion	Efficiency = 100% * (transferred - retransmitted) / transferred Network Loss = 100 - Efficiency
Transmission time	The time required for transmission of a message depends on the size of the message and the bandwidth of the channel.	Transmission time = Message size / Bandwidth
Availability	It is also known as uptime, simply measures whether or not the network is currently operational. You can never guarantee 100% availability, but you want to be aware of any downtime that happens on your network that you weren't expecting. It's important to be alerted when the network goes down, which network monitoring tools will provide for you. However, you should also be able to discover your actual uptime percentage and how often your network goes down.	
Connectivity	Connectivity refers to whether or not the connections between the nodes on your network are working properly. If there is an improper or malfunctioning connection on your network, it can be a major hurdle for your company. Ideally, every connection should be operating at peak levels at all times. However, performance issues like malware can target specific nodes or connections to affect performance in that specific area of the network.	

Retransmission	**When packets are lost, the network needs to retransmit it to complete a data request. This retransmission rate lets your enterprise know how often packets are being dropped, which is an indication of congestion on your network. You can analyse retransmission delay, or the time it takes for a dropped packet to be retransmitted, to understand how long it takes your network to recover from packet loss.**	
Propagation Time	**Propagation time measures the time required for a bit to travel from the source to the destination. The propagation time is calculated by dividing the distance by the propagation speed.**	**Propagation time = Distance / Propagation speed**
Processing Delay	**Time taken by the processor to process the data packet is called processing delay.**	
Queuing Delay	**Time spent by the data packet waiting in the queue before it is taken for execution is called queuing delay.**	
Jitter	**Jitter is defined as the variation in time delay for the data packets sent over a network. This variable represents an identified disruption in the normal sequencing of data packets. Jitter is related to latency, since the jitter manifests itself in increased or uneven latency between data packets, which can disrupt network performance and lead to packet loss and network congestion. Although some level of jitter is to be expected and can usually be tolerated, quantifying network jitter is an important aspect of comprehensive network**	**Latency = sum of all delays** **To measure Jitter, we take the difference between samples, then divide by the number of samples (minus 1).**

File Handling Operations using Socket Programming

Name	Functions
Mahima	1.Connect server to all clients using the application 2.Entering new client details if any 3.Viewing details of all existing clients
Darshan	1.Calculate Simple Interest 2.Gift Rewards based on usage.
Mathan	1.Collects the feedback given by the clients 2.Client - Server chat application.

Server Code:

```
import socket
import threading
import random
import string
import pandas as pd
import numpy as np
df1 = pd.DataFrame(pd.read_csv("case_study.csv"))
df1.set_index(['Customer_ID','Name',"Feedback"])

l=[]


def insert(clientsocket,df_df):
    data=[]
    message1="Enter Customer ID"
    clientsocket.send(message1.encode())
    id= clientsocket.recv(1024).decode()
    data.append(id)
    message1="Enter Name"
    clientsocket.send(message1.encode())
    cname= clientsocket.recv(1024).decode()
    data.append(cname)
    message1 = "Enter Feedback"
    clientsocket.send(message1.encode())
    cp = clientsocket.recv(1024).decode()
    data.append(cp)
    l=len(df_df)
    df_df.loc[l] = data
    j = df_df.to_string()
    clientsocket.send(j.encode())
    df1.to_csv('case_study.csv', index=False)

def chat(clientsocket):
    name = "Assistant"
    clientsocket.send(name.encode())
    s_name = clientsocket.recv(1024).decode()
    print(s_name, "has connected to the chat room\nEnter // to exit chat room\n")

    while True:
        message = input(str("Me : "))
        if message == "//":
            message = "Left chat room!"
            clientsocket.send(message.encode())
            print("\n")
            break
        clientsocket.send(message.encode())
        message = clientsocket.recv(1024)
        message = message.decode()
        print(s_name, ":", message)

def on_new_client(clientsocket,addr,host,df1):
```

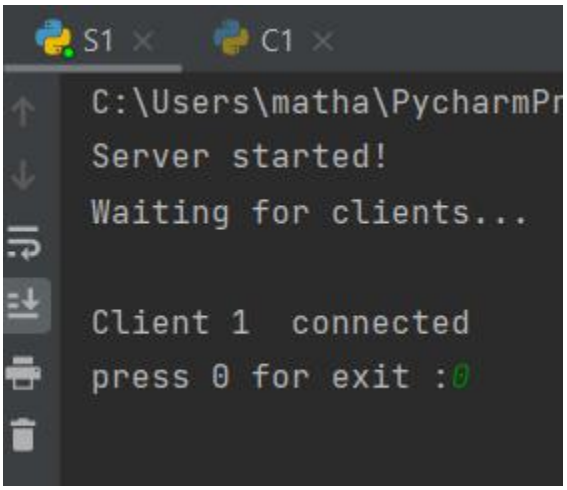
```
    msg = clientsocket.recv(1024).decode()
    print(msg," connected")
    message="Enter options\n----------\n1.Feedback\n2.Chat with server"
    clientsocket.send(message.encode())
    msg1 = clientsocket.recv(1024).decode()
    i =int(msg1)
    if i == 1:
        insert(clientsocket,df1)
    elif i == 2:
        chat(clientsocket)
    else:
        a="Trouble in server! Please try again later!"
        clientsocket.send(message.encode())

    clientsocket.close()


s = socket.socket()        # Create a socket object
host = socket.gethostname() # Get local machine name
port = 5001                # Reserve a port for your service.
print('Server started!')
print('Waiting for clients...\n')

s.bind((host, port))       # Bind to the port
s.listen(5)
j=True

while j:
    c,addr = s.accept()  # Establish connection with client.
    t = threading.Thread(target=on_new_client, args=(c, addr, host,df1))  # df[0:5] #########
    t.start()
    t.join()
    e=input("press 0 for exit :")
    if e==0:
        j=False
    else:
        pass

s.close()
```

Client Code:

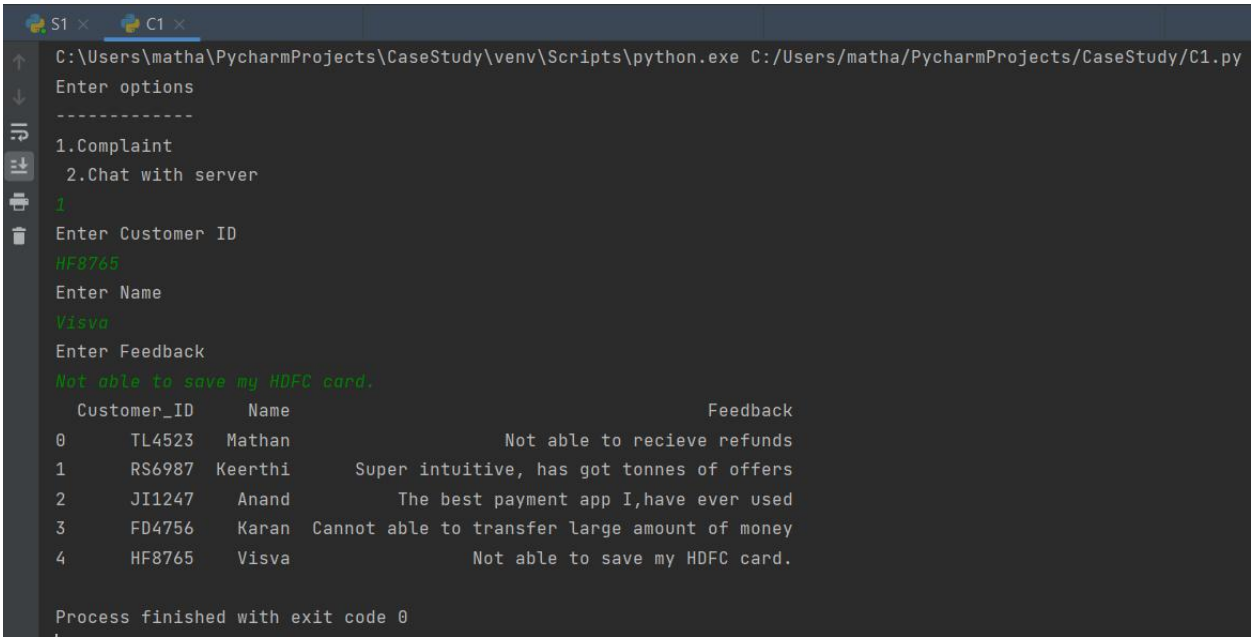
```
import socket
import threading

s = socket.socket()
host = socket.gethostname()
port = 5001

s.connect((host, port))
s.send(bytes("Client 1","utf-8"))

print(s.recv(1024).decode())
i = input()
```

```
if i=="1":
    s.send(bytes(i, "utf-8"))
    k1 = s.recv(1024).decode()
    print(k1)
    id = input()
    s.send(bytes(id, "utf-8"))
    k1 = s.recv(1024).decode()
    print(k1)
    name = input()
    s.send(bytes(name, "utf-8"))
    k1 = s.recv(1024).decode()
    print(k1)
    cp = input()
    s.send(bytes(cp, "utf-8"))
    k1 = s.recv(1024).decode()
    print(k1)
elif i=="2":
    s.send(bytes(i, "utf-8"))
    s_name = s.recv(1024).decode()
    name = str(input("Enter name:"))
    s.send(name.encode())

    print(s_name, "has joined the chat room\nEnter // to exit chat room\n")
    while True:
        message = s.recv(1024).decode()
        print(s_name, ":", message)
        message = input(str("Me : "))
        if message == "//":
            message = "Left chat room!"
            s.send(message.encode())
            print("\n")
            break
        s.send(message.encode())
s.close()
```

Server Output:

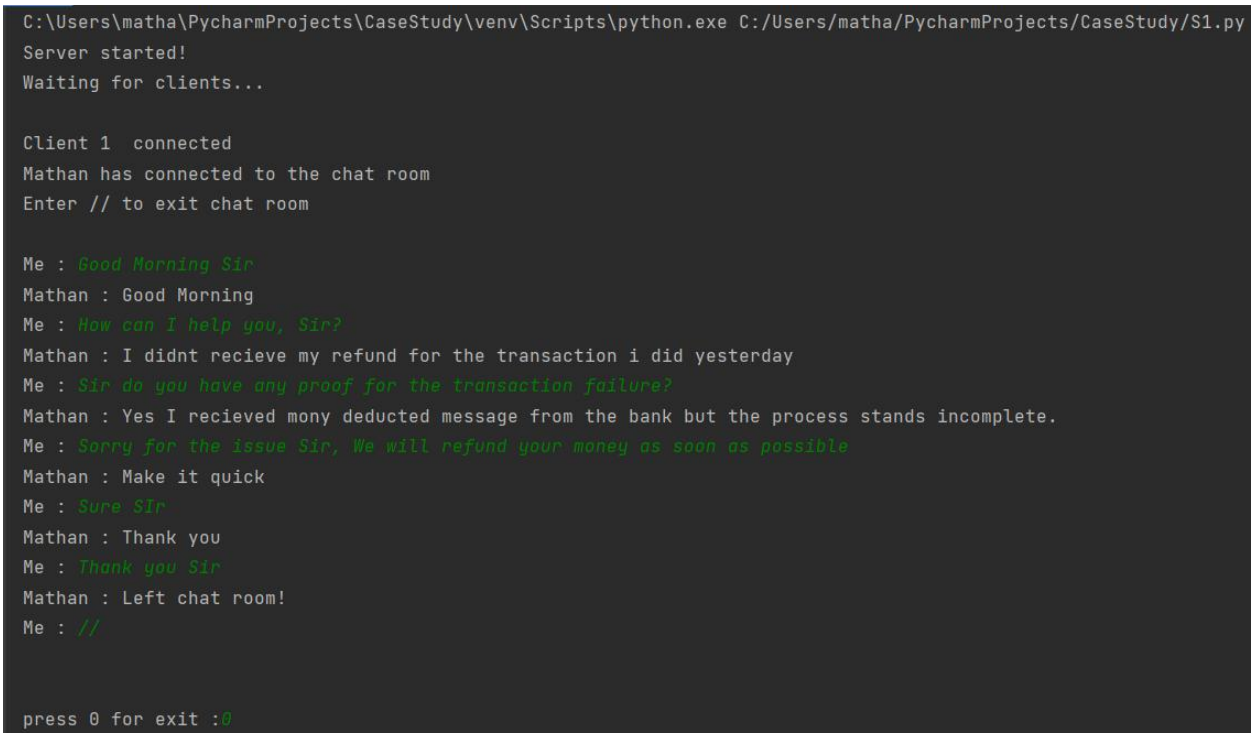


Client Output:



Chat Application:

Server Output:



Client Output:

```
C:\Users\matha\PycharmProjects\CaseStudy\venv\Scripts\python.exe C:/Users/matha/PycharmProjects/CaseStudy/C1.py
Enter options
-------------
1.Feedback
2.Chat with server
2
Enter name:Mathan
Assistant has joined the chat room
Enter // to exit chat room

Assistant : Good Morning Sir
Me : Good Morning
Assistant : How can I help you, Sir?
Me : I didnt recieve my refund for the transaction i did yesterday
Assistant : Sir do you have any proof for the transaction failure?
Me : Yes I recieved mony deducted message from the bank but the process stands incomplete.
Assistant : Sorry for the issue Sir, We will refund your money as soon as possible
Me : Make it quick
Assistant : Sure SIr
Me : Thank you
Assistant : Thank you Sir
Me : //



Process finished with exit code 0
```