

# **19CSE301 Computer Networks**

## **Case Study Submission**

**Date: 22.11.2021**

**Title: Online Bakery Delivery Portal**

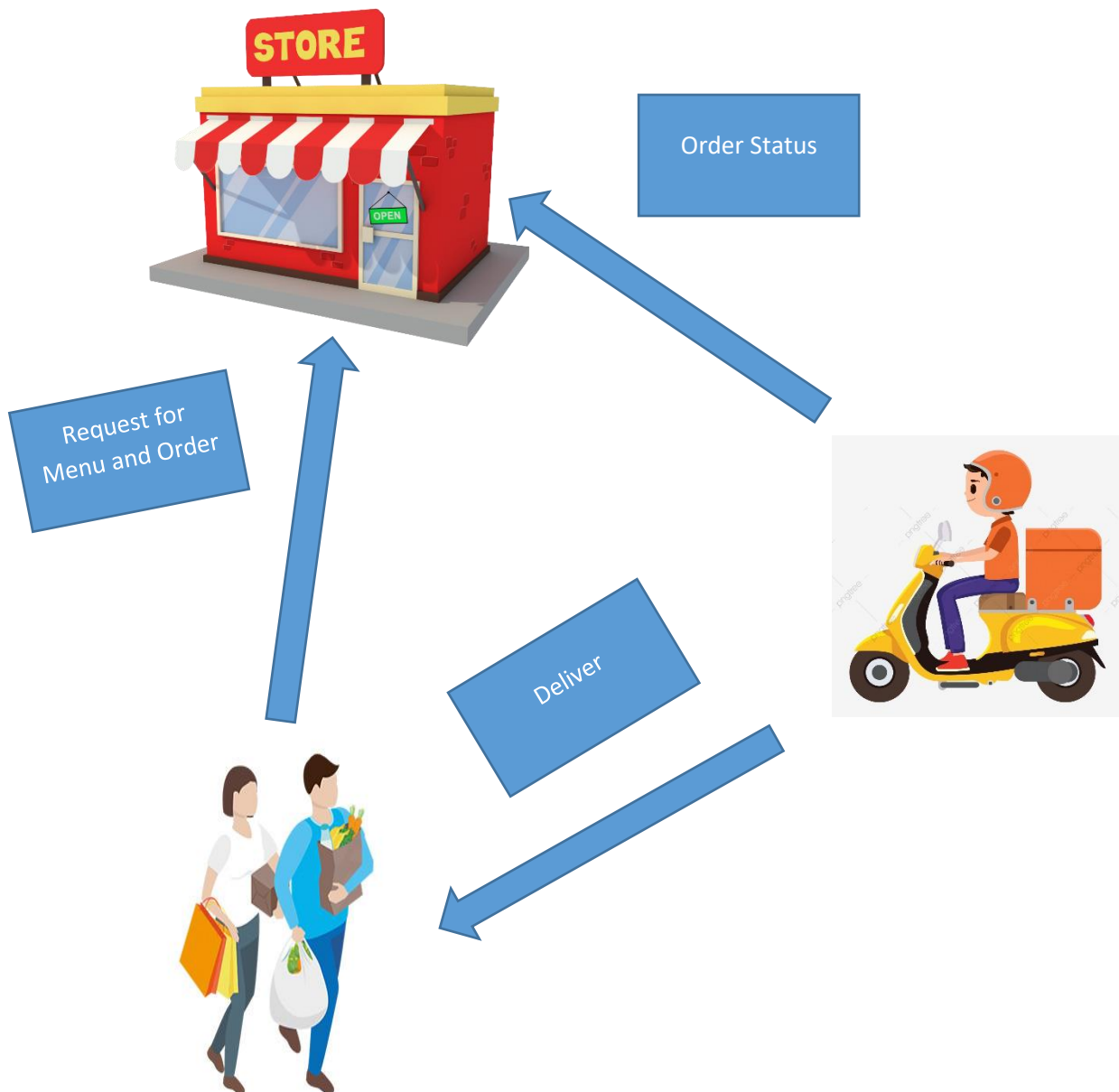
**Submitted by:**

<b>Roll No</b>	<b>Name</b>
CB.EN.U4CSE19301	Adheena B
CB.EN.U4CSE19341	Oviya B
CB.EN.U4CSE19353	Swathi P
CB.EN.U4CSE19364	Nivedya S

**Faculty:** Ms. Abirami K

### Description of the case study:

This case study aims to look into the structure of an online order-delivery portal for a small business (Bakery). The network provides various services for customers as well as to the workers of the business. This network will allow the usage of technology and automate the various underlying processes, making it accessible for people from anywhere by using any device that they have.



### Why is Networking required?

Computer networks help users on the network to share the resources, exchange information and communicate with one another.

Using a network, we can:

- Remotely access documents or applications stored across different systems.
- Store and manipulate data across different systems connected in the network.
- Communicate with one another through any device i.e., through emails, video calls etc.
- Setup peripherals or hardware devices like printers, hard drives, scanners for multiple devices in the same network.

### Why do we require Networking in our application?

We can establish efficient and accurate communication between various sections/departments of the Bakery i.e.,

- customer-product interactions,
- delivery – dispatch details,
- worker logs etc.

It is essential to preserve the speed, integrity and security of the information passed between the departments. Failure to do so will lead to financial losses and slower operations.

With the help of an effective network infrastructure the performance of the business will be increased along with customer satisfaction.

### File Handling Operations using Socket Programming

#### a. Data files:

##### 1. Order

	A	B	C	D	E	F
1	orderID	UserName	UserGmail	OrderTaken	Cost	DeliveryStatus
2	1	Aden	Aden@gmail.com	Cake-2,Burger-3		
3	2	Marsh	Marsh@gmail.com	Pizza-2,Burger-4		
4	3	Perkins	Perkins@gmail.com	Cake-2,Pizza-2		
5	4	Atkinson	Atkinson@gmail.com	Burger-2		
6	5	Genesis	Genesis@gmail.com	Pizza-3		
7	6	Giovanny	Giovanny@gmail.com	Cake-2		
8	7	Elena	Elena@gmail.com	Cake-2,Burger-2		
9	8	Dalton	Dalton@gmail.com	Cake-4		
10	9	Octavio	Octavio@gmail.com	Cake-2,Burger-1,Pizza-3		
11	10	Suresh	suresh@gmail.com	Cake-2,Burger-1	120	Order confirmed
12						
13						

## 2. Menu

	A	B
1	Item	Cost
2	Pizza	199
3	Burger	120
4	Cake	50
5	Smoothie	70
6	Cookies	30
7	Fresh juice	20
8	Tea	15
9	Coffee	10
10	MilkShake	25
11	Chips	35
12	French Fries	60
13	Coke	40
14	Veg Balls	70

## 3. Workers Sheet

WorkerID	Name	Gmail	WorkerPwd
591	John	<a href="mailto:John@gmail.com">John@gmail.com</a>	nhoj
592	Dao	<a href="mailto:Dao@gmail.com">Dao@gmail.com</a>	oad
593	Ken	<a href="mailto:Ken@gmail.com">Ken@gmail.com</a>	nek
594	Ekaterini	<a href="mailto:Ekaterini@gmail.com">Ekaterini@gmail.com</a>	teriniake
595	Jorge	<a href="mailto:Jorge@gmail.com">Jorge@gmail.com</a>	geroj
596	Lucas	<a href="mailto:Lucas@gmail.com">Lucas@gmail.com</a>	ascul
597	Thanos	<a href="mailto:Thanos@gmail.com">Thanos@gmail.com</a>	nosaht
598	Thor	<a href="mailto:Thor@gmail.com">Thor@gmail.com</a>	roht
599	Krick	<a href="mailto:Krick@gmail.com">Krick@gmail.com</a>	ckirk
600	Jim	<a href="mailto:Jim@gmail.com">Jim@gmail.com</a>	mij

### b. List of operations completed with the File

1. Viewing Menu of items
2. Take an Order
3. Generate Bill for the order

4. View Delivery Status of an order
5. Update delivery status of an order (workers)

**Socket Programming Code:**

### **Workers\_Client.py**

```
import socket

clientSocket = socket.socket()

try:
    clientSocket.connect((socket.gethostname(),1025))
except socket.error as err:
    print(str(err))

print(clientSocket.recv(1000).decode('utf-8'))

print(clientSocket.recv(1000).decode('utf-8'))
mailID = input()
clientSocket.send(mailID.encode('utf-8'))

print(clientSocket.recv(1000).decode('utf-8'))
WokerPwd = input()
clientSocket.send(WokerPwd.encode('utf-8'))
print(clientSocket.recv(1000).decode('utf-8'))

cont = "Yes"
while (cont == "Yes" or cont == "yes"):
    print(clientSocket.recv(1000).decode('utf-8'))
    OrderTaken = input()
    clientSocket.send(OrderTaken.encode('utf-8'))
    print(clientSocket.recv(1000).decode('utf-8'))
    DeliveryStatus = input()
    clientSocket.send(DeliveryStatus.encode('utf-8'))
    print(clientSocket.recv(1000).decode('utf-8'))
    cont = input()
    clientSocket.send(cont.encode('utf-8'))
```

### **Workers\_Server.py**

```
import socket
import pandas as pd
from _thread import start_new_thread

serverSocket = socket.socket()
```

```

noOfClients = 0
try:
    serverSocket.bind((socket.gethostname(), 1025))
except socket.error as err:
    print(str(err))

print("Waiting for connection...")
serverSocket.listen(8)

def servicesProvided(client):
    df = pd.read_excel('WorkersSheet.xlsx')
    client.send("Enter MAIL ID".encode('utf-8'))
    mail = client.recv(1000).decode('utf-8')
    client.send("Enter Password".encode('utf-8'))
    WorkerPwd = client.recv(1000).decode('utf-8')
    ans = df[(df['Gmail'] == mail) & (df['WorkerPwd'] == WorkerPwd)]
    if (ans.empty):
        client.send("Entered Gmail/Password is Invalid".encode('utf-8'))
        client.close()
    else:
        order = pd.read_csv('Order.csv')
        client.send("Access Granted...".encode('utf-8'))
        cont = "Yes"
        while (cont == "Yes" or cont == "yes"):
            client.send('Enter the Order number:'.encode('utf-8'))
            orderID = int(client.recv(1000).decode('utf-8'))
            client.send('Enter the delivery Status:'.encode('utf-8'))
            DeliveryStatus = client.recv(1000).decode('utf-8')
            order.set_index('orderID', inplace=True)
            order.loc[orderID, 'DeliveryStatus'] = DeliveryStatus
            order.reset_index(inplace=True)
            order.to_csv("Order.csv", index=False)
            client.send("Successful!\nDo you want to edit status of any more o
rder? Yes/No?".encode('utf-8'))
            cont = client.recv(1000).decode('utf-8')
        client.close()

while True:
    client, adrs = serverSocket.accept()
    client.send("Connecting to server...".encode('utf-8'))
    start_new_thread(servicesProvided, (client,))
    noOfClients += 1

```

## OUTPUTS

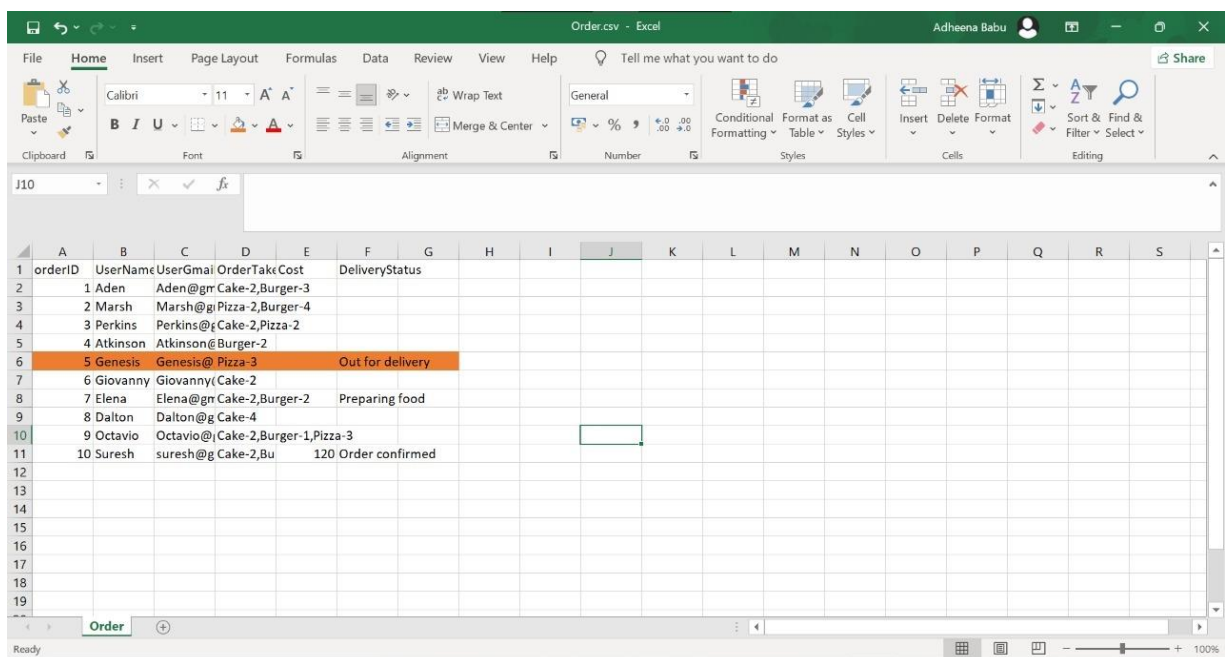
### Worker Client output

```
Worker_Server x Worker_Client x
C:\Users\hp\PycharmProjects\socketProgramming\venv\Scripts\python.exe
Connecting to server...
Enter MAIL ID
Lucas@gmail.com
Enter Password
oscul
Access Granted...
Enter the Order number:
5
Enter the delivery Status:
Out for delivery
Successful!
Do you want to edit status of any more order? Yes/No?
No
Process finished with exit code 0
```

### Worker Server output

```
Worker_Server x Worker_Client x
C:\Users\hp\PycharmProjects\socketProgramming\venv\Scripts\python.exe
Waiting for connection...
Process finished with exit code -1
```

### Change made by worker reflected in Order.csv file



orderID	UserName	UserGmail	OrderTake	Cost	DeliveryStatus
1	Aden	Aden@gm	Cake-2,Burger-3		
2	Marsh	Marsh@g	Pizza-2,Burger-4		
3	Perkins	Perkins@g	Cake-2,Pizza-2		
4	Atkinson	Atkinson@g	Burger-2		
5	Genesis	Genesis@	Pizza-3	Out for delivery	Out for delivery
6	Giovanny	Giovanny@	Cake-2		
7	Elena	Elena@gm	Cake-2,Burger-2	Preparing food	
8	Dalton	Dalton@g	Cake-4		
9	Octavio	Octavio@	Cake-2,Burger-1,Pizza-3		
10	Suresh	suresh@g	Cake-2,Bu	120	Order confirmed

## Customer\_Client.py

```
import socket

clientSocket = socket.socket()

try:
    clientSocket.connect((socket.gethostname(), 1024))
except socket.error as err:
    print(str(err))

print(clientSocket.recv(1000).decode('utf-8'))
print("Enter Name: ")
name = input()
clientSocket.send(name.encode('utf-8'))
print(clientSocket.recv(1000).decode('utf-8'))
mailID = input()
clientSocket.send(mailID.encode('utf-8'))

home = "Yes"
while (home == "yes" or home == "Yes"):
    print(clientSocket.recv(1000).decode('utf-8'))
    option = input()
    clientSocket.send(option.encode('utf-8'))
    option = int(option)
    if (option == 1):
        print(clientSocket.recv(1000).decode('utf-8'))
        orderID = input()
        clientSocket.send(orderID.encode('utf-8'))
        print(clientSocket.recv(1000).decode('utf-8'))
    elif (option == 2):
        continueprocess = 'Yes'
        while (continueprocess == 'Yes'):
            print(clientSocket.recv(1000).decode('utf-8'))
            print(clientSocket.recv(1000).decode('utf-8'))
            orderTaken = input()
            clientSocket.send(orderTaken.encode('utf-8'))
            print(clientSocket.recv(1000).decode('utf-8'))
            print(clientSocket.recv(1000).decode('utf-8'))
            print(clientSocket.recv(1000).decode('utf-8'))
            continueprocess = input()
            clientSocket.send(continueprocess.encode('utf-8'))
        print(clientSocket.recv(1000).decode('utf-8'))
    else:
        print(clientSocket.recv(1000).decode('utf-8'))
        break
print('Do you want to go to home screen?')
home = input()
clientSocket.send(home.encode('utf-8'))
```



## Customer\_Server.py

```
import socket
import pandas as pd
from _thread import start_new_thread

serverSocket = socket.socket()
noOfClients = 0
try:
    serverSocket.bind((socket.gethostname(),1024))
except socket.error as err:
    print(str(err))

print("Waiting for connection...")
serverSocket.listen(8)

def servicesProvided(client):
    df = pd.read_csv('Order.csv')
    name = client.recv(1000).decode('utf-8')
    print(name,"is online")
    client.send("Enter MAIL ID".encode('utf-8'))
    mail = client.recv(1000).decode('utf-8')
    home = "yes"
    while(home == "yes" or home == "Yes"):
        intro = '''Choose an option:
            1)View status of an order
            2)Make a New order
            ...
        client.send(intro.encode('utf-8'))
        opt = client.recv(1000).decode('utf-8')
        opt = int(opt)
        if (opt==1):

            client.send("Enter order ID : ".encode('utf-8'))
            pid = client.recv(1000).decode('utf-8')
            print(pid)
            info = df[df['orderID'] == int(pid)]
            info = info.to_string(index=False)
            requestedData = info.encode('utf-8')
            client.send(requestedData)
        elif (opt==2):
            menu = pd.read_csv('menu.csv')
            menuData = menu.to_string(index=False)
            client.send(menuData.encode('utf-8'))
            cont = "Yes"
            total = 0
            df1 = pd.DataFrame(columns=['Item', 'Quantity'], index=None)
            finalOrder = ""
            while (cont == "Yes"):
```

```

        client.send("Enter your order from the menu in the format ITEM
-QUANTITY separated by comma(,)".encode('utf-8'))
        order = client.recv(1000).decode('utf-8')
        finalOrder+=order
        client.send("Order being processed...".encode('utf-8'))

        for m in order.split(","):
            item, quantity = m.split("-")
            quantity = int(quantity)
            itemlist = menu[menu['Item'] == item]
            cost = itemlist.iloc[0]['Cost']
            df2 = {'Item': item, 'Quantity': quantity}
            df1 = df1.append(df2, ignore_index=True)
            total += cost * quantity
            msg = "Order processed successfully....\n"
            msg += df1.to_string(index=False)
            client.send(msg.encode('utf-8'))
            client.send("Would you like to order anything else?".encode('u
tf-8'))

            cont = client.recv(1000).decode('utf-8')
            count = df.iloc[-1,0]
            orderentry = {'orderID':count+1, 'UserName': name, 'UserGmail': ma
il, 'OrderTaken': finalOrder, 'Cost': total, 'DeliveryStatus': "Order confirme
d"}

            df = df.append(orderentry, ignore_index=True)
            df.to_csv("Order.csv",index=False)

            bill = ''
            BILL
            -----\n'''

            bill += df1.to_string(index=False)
            bill += "\nGrand Total : " + str(total) + "\nOrder Placed Successf
ully.\nOrder Number : " + str(count+1)
            client.send(bill.encode('utf-8'))

            file = name + ".txt"
            file1 = open(file, 'a+')
            file1.write(bill)
            file1.close()

        else:
            client.send("Invalid option....closing connection".encode('utf
-8'))

            client.close()
            home = client.recv(1000).decode('utf-8')
            client.close()

```

```

while True:
    client, adrs = serverSocket.accept()
    client.send("Connecting to server...".encode('utf-8'))
    start_new_thread(servicesProvided,(client,))
    noOfClients += 1

```

## OUTPUTS:

### Customer Client output

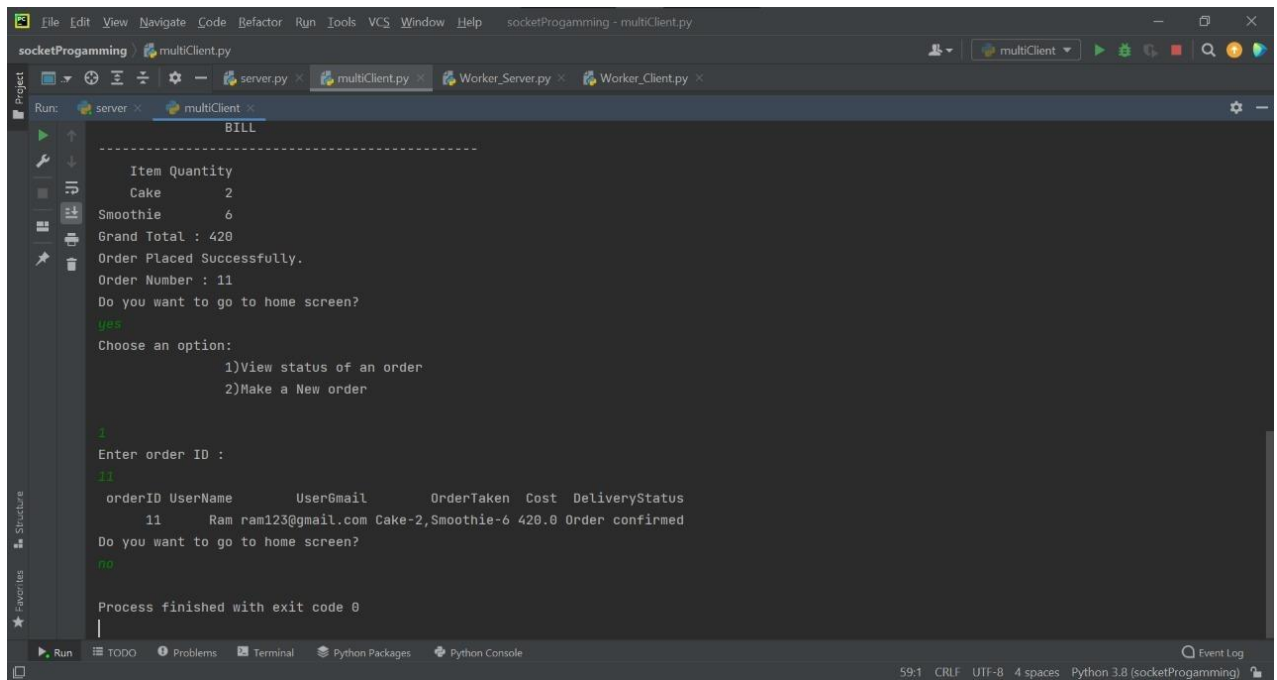
The first screenshot shows the initial output of the multi-client program. It displays a menu with the following items and costs:

Item	Cost
Pizza	199
Burger	120
Cake	50
Smoothie	70
Cookies	30
Fresh juice	20
Tee	15
Coffee	10
MilkShake	25
Chips	35
French Fries	60
Coke	40

The second screenshot shows the order processing flow. The user enters their order in the format ITEM-QUANTITY separated by comma(,). The order is processed successfully, and a bill is generated. The bill shows the following items and quantities:

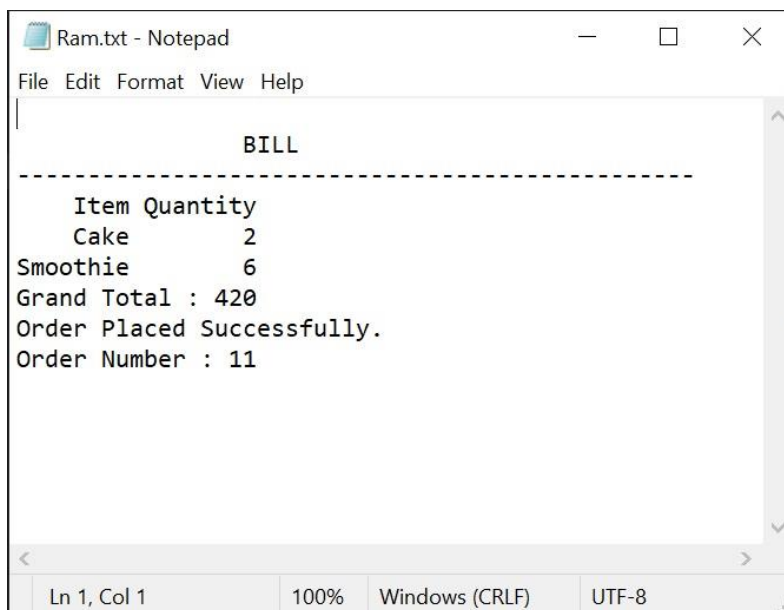
Item	Quantity
Cake	2
Smoothie	6

The Grand Total is 420. The order is placed successfully, and the order number is 11. The user is asked if they want to go to the home screen, and they respond 'yes'. The program then displays the same menu as in the first screenshot.



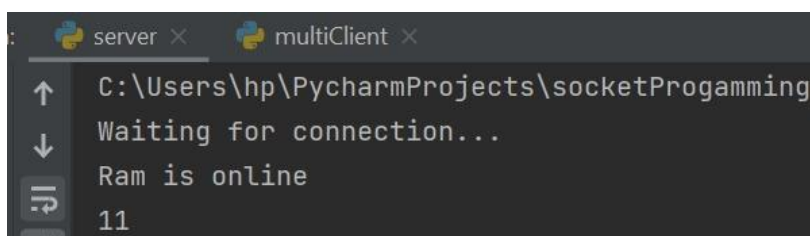
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help socketProgramming - multiClient.py
socketProgramming multiClient.py
Run: server multiClient.py Worker_Server.py Worker_Client.py
Run: server multiClient.py
BILL
-----
Item Quantity
Cake 2
Smoothie 6
Grand Total : 420
Order Placed Successfully.
Order Number : 11
Do you want to go to home screen?
yes
Choose an option:
1)View status of an order
2)Make a New order
1
Enter order ID :
11
orderID UserName UserGmail OrderTaken Cost DeliveryStatus
11 Ram ram123@gmail.com Cake-2,Smoothie-6 420.0 Order confirmed
Do you want to go to home screen?
no
Process finished with exit code 0
```

## **Bill Generated:**



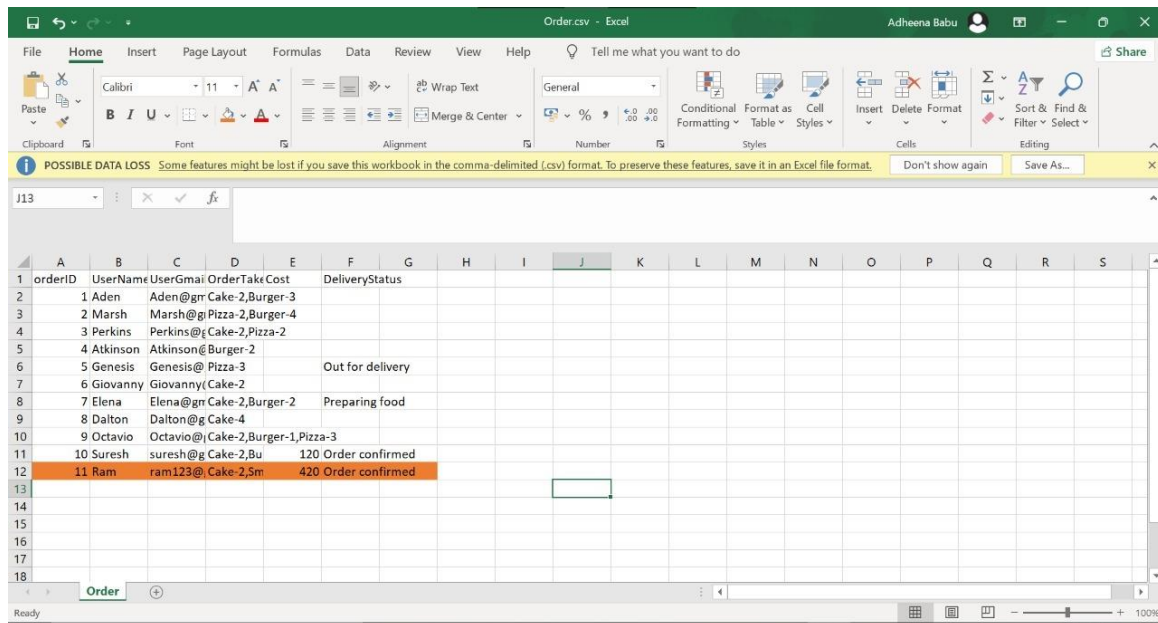
```
Ram.txt - Notepad
File Edit Format View Help
BILL
-----
Item Quantity
Cake 2
Smoothie 6
Grand Total : 420
Order Placed Successfully.
Order Number : 11
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

## **Customer Server Output**



```
server multiClient
C:\Users\hp\PycharmProjects\socketProgramming
Waiting for connection...
Ram is online
11
```

## Ram's order reflected in Order.csv file



orderID	UserName	UserGmail	OrderTaki	Cost	DeliveryStatus
1	Aden	Aden@gm	Cake-2,Burger-3		
2	Marsh	Marsh@gi	Pizza-2,Burger-4		
3	Perkins	Perkins@t	Cake-2,Pizza-2		
4	Atkinson	Atkinson@	Burger-2		
5	Genesis	Genesis@	Pizza-3		Out for delivery
6	Giovanny	Giovanny@	Cake-2		
7	Elena	Elena@gm	Cake-2,Burger-2		Preparing food
8	Dalton	Dalton@g	Cake-4		
9	Octavio	Octavio@i	Cake-2,Burger-1,Pizza-3		
10	Suresh	suresh@g	Cake-2,Bu	120	Order confirmed
11	Ram	ram123@	Cake-2,\$m	420	Order confirmed

## Cisco Packet Tracer:

### a. Description:

#### Application Protocols used:

1. SMTP:
  - I. Transfers the mail from Customers to server.
  - II. Transfers the mail from Server to Customers.
2. DNS:
  - I. To Access the website both from customer side and worker side.
  - II. Link: [www.strivers.com](http://www.strivers.com)
3. FTP:
  - I. DeliveryStatus.txt contains the status of the order.
  - II. Transfer of the file from Server to Customer.

#### Routing Protocols:

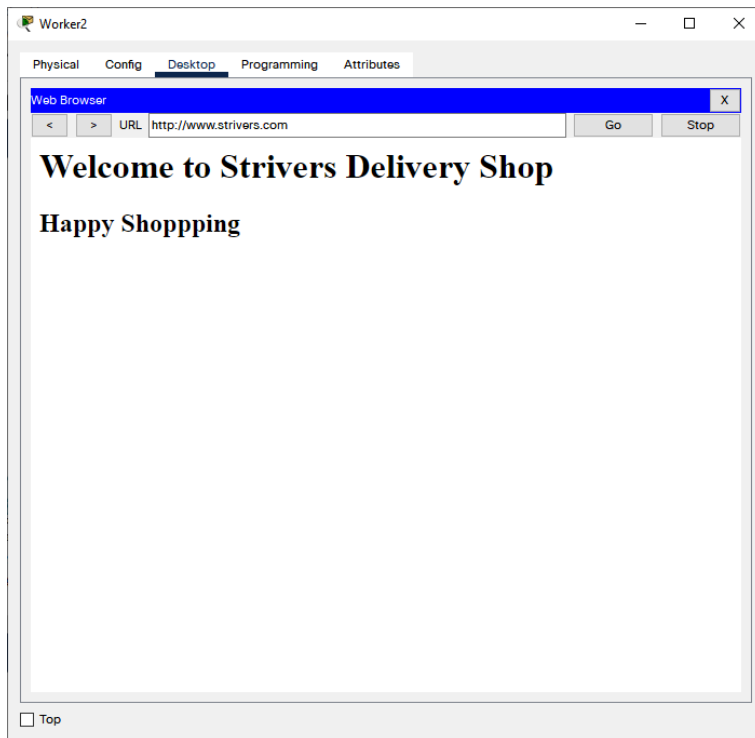
##### OSPF:

OSPF protocol has no limitations in hop count, unlike RIP protocol that has only 15 hops at most. So OSPF converges faster than RIP and has better load balancing.

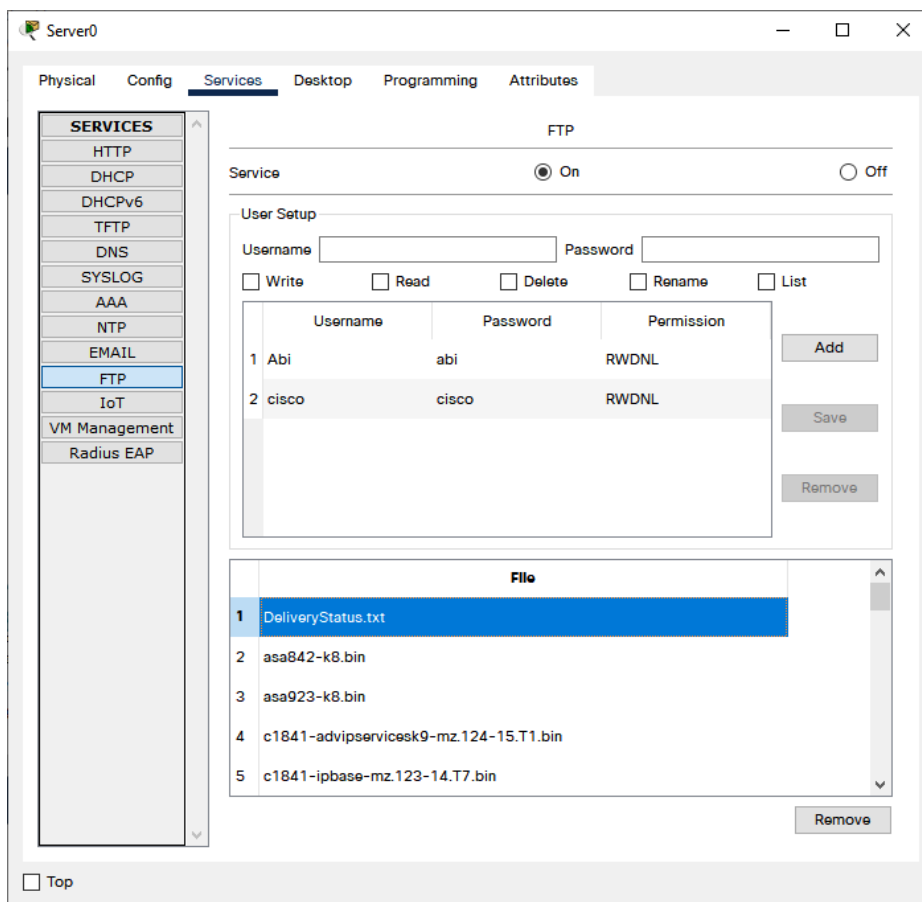
Used VLAN for restricting access to Interns working in the shop.

## b. Simulation Output:

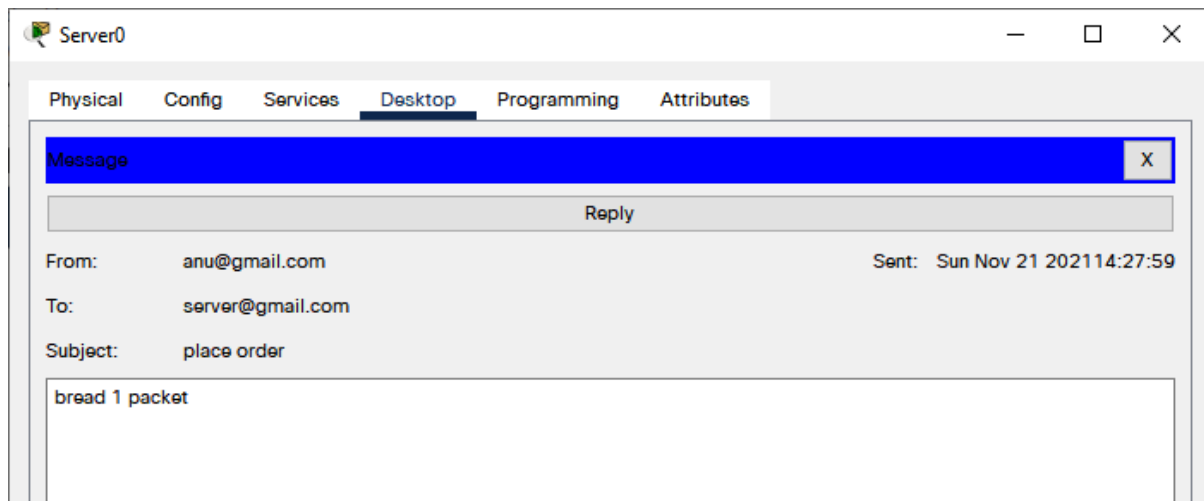
### Website Output DNS:



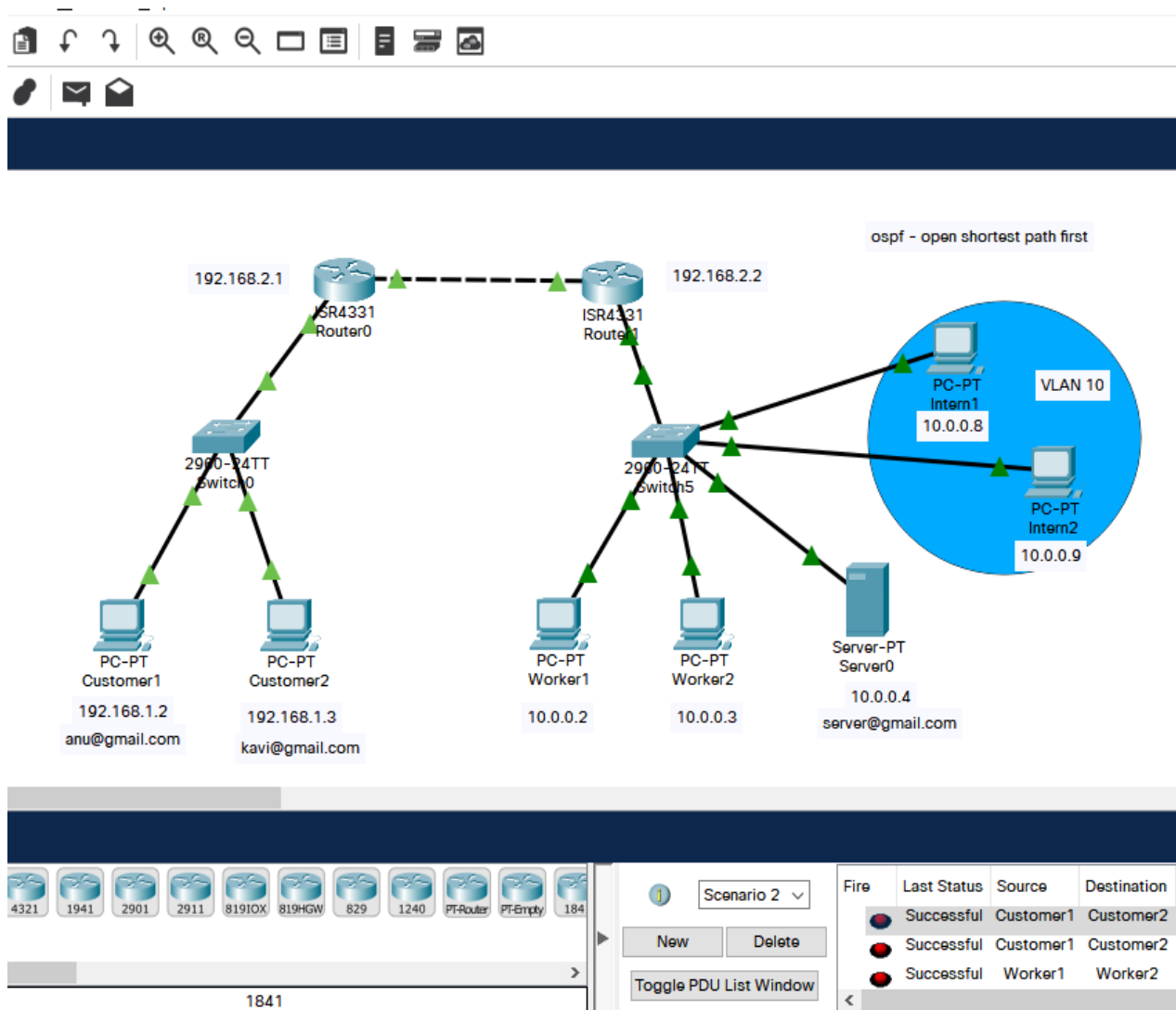
### FTP:



## SMTP mail:



## Overall Configuration:



## **Cloud Computing and Virtualization:**

Cloud is considered to be the cheapest, fastest and easy-to-use technology. It brings down the hardware and software demand by shifting the workload into other systems. It supports multi-tenancy i.e., sharing of resources among two or more clients and Virtualization.

A cloud system will store all the data over the internet rather than a physical storage drive. In an event where the local hardware crashes the data is backed up through as well as we can reboot the system by allocating the machine to another host/resources through virtualization.

Cloud Computing will provide our existing network:

- IT flexibility,
- Lesser expenditure on hardware costs,
- Larger scalability,
- Higher Computing power with additional resources.

## **Cloud Model for our Application: Hybrid cloud model**

Our application can take up the hybrid cloud model as it can integrate combination of a private cloud for handling sensitive data like billing information containing credit card details, customer personal information, revenue information which requires high security compliance and for less critical data and front-end customer interactions, a public cloud. This will give us the advantage of gaining better performance alongside lower price by eliminating physical, on-premise servers and associated maintenance costs.

**Cloud service opted:** SaaS or Software as a Service e.g. Cisco WebEx, BigCommerce

SaaS can also be called as “pay-as-you-go application”. As the operations handled by our portal is User Authentication, Invoice Generation and Worker Management. Using SaaS is the best option as,

- Low Set-Up: The SaaS services are easy to buy and has Low maintenance costs.
- Less Operation Complexity: Easy to use UI, the minimum specification for the user is any device which has an internet connection.
- Minimization Hardware Requirement: SaaS are hosted remotely, therefore hardware requirements are minimum.

As our application is a Bakery Order portal which may/may not have an IT specialist. Going with SaaS is the better option as setting up is easier than PaaS or IaaS as well as the paid packages of SaaS is inclusive of security, compliance, and maintenance as part of the cost.

## **Virtualization:**

Virtualization plays a very important role in the cloud computing technology, normally in the cloud computing, users share the data present in the clouds like application etc., but actually with the help of virtualization users shares the infrastructure.

The scope of Cloud Virtualization for our application extends over:

### **1. Storage Virtualization:**

- a. File Servers are a good addition into the model where the data can be stored remotely.



- b. Data is stored in the more convenient locations away from the specific host. In the case of a host failure, the data is not compromised necessarily.
- c. It can provide easy backup, and recovery of data. the data is stored in different places with maximum security. If any disaster to the local storage takes place the data can be retrieved from some other place and it won't affect the customer.

## **2. Server Virtualization: With a Type 1 Hypervisor**

- a. When the virtual machine software is directly installed on the Server system is known as server virtualization.
- b. Server Virtualization will allow the main physical server to be rationed into multiple independent virtual private servers as per need. This will allow for efficient load balancing
- c. Type 1 Hypervisor in para virtualization provides lesser latency as there is no OS required and no emulation overhead as the hypervisor doesn't use the allocated resources for its own needs. Therefore. Increasing the performance.