

23rd November, 2021

19CSE301

Computer Networks

CASE STUDY



Carpooling System

GROUP - 5

Team members:

Kritika M A

CB.EN.U4CSE19629

S Sai Chandra Kaushik

CB.EN.U4CSE19651

Leepaakshi G

CB.EN.U4CSE19630

D Rhuthvik

CB.EN.U4CSE19614

Abstract

Carpooling is a transport system based on a shared use of private cars. Carpooling reduces each person's travels costs such as fuel costs, tolls, and the stress of driving. Carpooling is one method that can be easily instituted and can help resolve a variety of problems that continue to plague urban areas, ranging from energy demands and traffic congestion to environmental pollution. The Carpool system would enable its user a safe and secure way to share cars. This could include both short daily journeys such as going to workplace within the city, long inter-city trips and outstation trips.

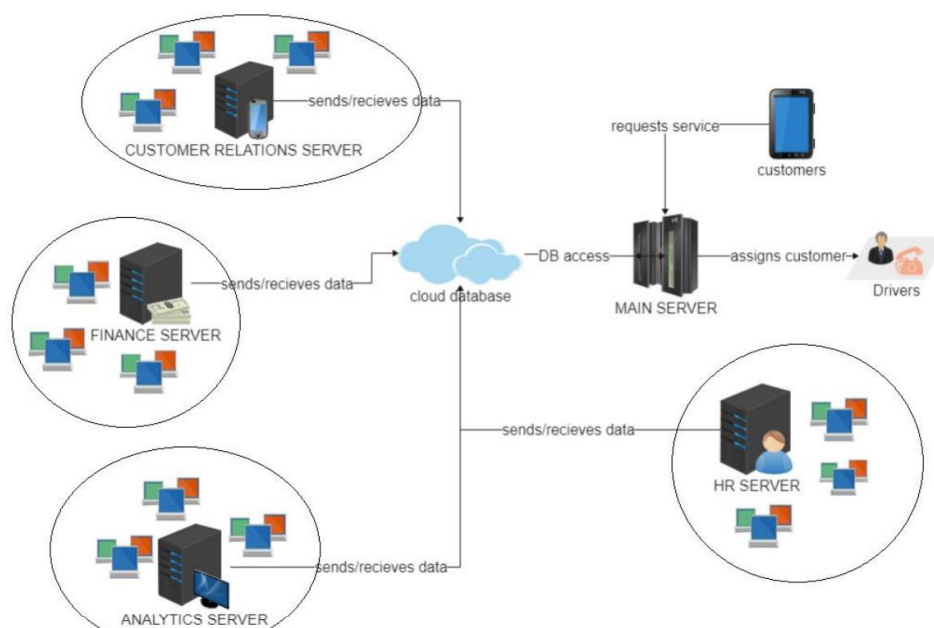
A Carpooling system is a system that, via websites and mobile apps, matches passengers with drivers of vehicles for hire

Working:

- collects location, name, and a few other metadata (a set of data that gives information about other data) from the rider and driver client
- using that data, the system syncs the rider and driver client so that they exchange current location data between themselves which enables them to share a ride.

The driver client sends his/her location to the app server every minute or every second. The rider client will need to send a HTTP request to get the driver's location from the server every minute or second. Accuracy is based on the time interval between the requests the lesser the time the more the accuracy.

Architecture Diagram



Why networking is required?

While the entire working of a carpooling system depends on computer networks for connecting the customers to the riders, the benefit of a two-sided network is evident in these applications. Two-sided network benefit refers to the increase in usage of a product by one set of users.

Carpooling system will provide a sufficient number of drivers as to lower the potential waiting times for their respective clients – the riders. With the ample supply of drivers and minimal waiting times the drivers will be able to complete more trips which in turn leads to increased revenues. Due to the high ride rate and potential to increase ones earning, this system will encourage more people to become drivers for the company. As one can see if system remains as such, it is favorable for the riders in terms of short waiting times and low costs and there will be a higher demand for drivers – these drivers would be incentivized to join the company as a result of the possibility of a higher earning potential created by the increased demand for rides.

SOCKET PROGRAMMING:

Working on generated log files to analyse patterns such as impact on revenue by updating pricing on a certain date, finding number of each type of rides taken etc.

The working excel will contain the following elements which will be the transaction values between the entities (driver and passengers)

Column name	Purpose
Ride id	Unique ID to identify a ride
Date	Date of the day of the ride
Customer Name	Name of the customer
Driver Id	Unique ID to identify registered drivers
Type	Instation or Outstation
Distance (in km)	Distance covered during the ride
Fare per km	Price of each kilometre
Base fare	The base pay over which the fare per km is applied
Total cost	Base Fare + (Distance * Fare per km)

List of Operations:

- 1.Insert New Record
2. Update Base Fare based on Date

3. Update Fare per km based on Date
4. Fetch all records based on Date
5. Fetch all records based on Driver ID
6. Update Total Cost = Base fare+ (fare per km*distance)
7. Print no.of each type of rides
8. View all

Server-side code :

```
import socket
from _thread import *
import os
import pandas as pd
import openpyxl

ServerSideSocket = socket.socket()
host = '127.0.0.1'
port = 2004
ThreadCount = 0
try:
    ServerSideSocket.bind((host, port))
except socket.error as e:
    print(str(e))

print('Socket is listening..')
ServerSideSocket.listen(5)

def multi_threaded_client(connection):
    connection.send(str.encode('Server is working:'))
    while True:
        msg = connection.recv(2048).decode('utf-8')
        data = msg.split('%')
        df = pd.read_excel("logs.xlsx")
        data_dict = df.to_dict(orient='list')

        if (data[0] == 'I'):
            data_dict['Ride ID'].append(data[1])
            data_dict['Date'].append(data[2])
            data_dict['Customer Name'].append(data[3])
            data_dict['Driver ID'].append(data[4])
            data_dict['Type'].append(data[5])
            data_dict['Distance(in km)'].append(float(data[6]))
```

```

        data_dict['Fare per km'].append(float(data[7]))
        data_dict['Base Fare'].append(float(data[8]))
        data_dict['Total Cost'].append(0)
        new = pd.DataFrame.from_dict(data_dict, orient='index').transpose(
    )

    new.to_excel("logs.xlsx", index=False)
    send = "Successfully Inserted"

elif (data[0] == 'UBF'):
    temp = data_dict['Date']
    for j in range(len(temp)):
        if temp[j] == data[1]:
            data_dict['Base Fare'][j] = float(data[2])
    new = pd.DataFrame.from_dict(data_dict, orient='index').transpose(
    )

    new.to_excel('logs.xlsx', index=False)
    send = "Successfully Updated"

elif (data[0] == 'UF'):
    temp = data_dict['Date']
    for j in range(len(temp)):
        if temp[j] == data[1]:
            data_dict['Fare per km'][j] = float(data[2])
    new = pd.DataFrame.from_dict(data_dict, orient='index').transpose(
    )

    new.to_excel("logs.xlsx", index=False)
    send = "Successfully Updated"

elif (data[0] == 'FD'):
    send = ""
    temp = data_dict['Date']
    lis = []
    for i in range(len(temp)):
        if temp[i] == data[1]:
            lis.append(i)
    for j in data_dict.keys():
        send += j
        send += "*"
        for k in lis:
            send += str(data_dict[j][k])
            send += "%"
        send += "*"

elif (data[0] == 'FDr'):
    send = ""
    temp = data_dict['Driver ID']
    lis = []
    for i in range(len(temp)):

```

```

        if temp[i] == data[1]:
            lis.append(i)

    for j in data_dict.keys():
        send += j
        send += "*"
        for k in lis:
            send += str(data_dict[j][k])
            send += "%"
        send += "*"

    elif (data[0] == "TC"):
        for j in range(len(data_dict['Ride ID'])):
            data_dict['Total Cost'][j] = data_dict['Base Fare'][j] + (
                data_dict['Fare per km'][j] * data_dict['Distance(
in km)'][j])
        new = pd.DataFrame.from_dict(data_dict, orient='index').transpose(
)

        new.to_excel("logs.xlsx", index=False)
        send = "Successfully Updated"

    elif (data[0] == "T"):
        temp = data_dict['Type']
        send = ""
        In = 0
        Out = 0
        for j in range(len(temp)):
            if temp[j] == "Instation":
                In += 1
            elif temp[j] == "Outstation":
                Out += 1
        send += "Instation"
        send += "%"
        send += str(In)
        send += "%"
        send += "Outstation"
        send += "%"
        send += str(Out)

    elif (data[0] == "V"):
        send = ""
        for j in data_dict.keys():
            send += j
            send += "*"
            for k in data_dict[j]:
                send += str(k)
                send += "%"
            send += "*"

```

```

        if not data:
            break
        connection.sendall(str.encode(send))
    connection.close()

while True:
    Client, address = ServerSideSocket.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(multi_threaded_client, (Client,))
    ThreadCount += 1
    print('Thread Number: ' + str(ThreadCount))

ServerSideSocket.close()

```

Client-side code

```

import socket
import pandas as pd

ClientMultiSocket = socket.socket()
host = '127.0.0.1'
port = 2004
print('Waiting for connection response')
try:
    ClientMultiSocket.connect((host, port))
except socket.error as e:
    print(str(e))

res = ClientMultiSocket.recv(1024)
while True:
    print(" I - Insert New Record")
    print(" UBF - Update Base Fare based on Date")
    print(" UF - Update Fare per km based on Date")
    print(" FD - Fetch all records based on Date")
    print(" FDr - Fetch all records based on Driver ID")
    print(" TC - Update Total Cost = Base fare+(fare per km*dist)")
    print(" T - Print no.of each type of rides")
    print(" V - View all")
    Input = input("Choose from the options: ")
    data = Input
    data += '%'
    if (Input == 'I'):
        print("Enter Ride ID:", end="")
        temp = input()
        temp += '%'
        data += temp

```

```

print("Enter Date:", end="")
temp = input()
temp += '%'
data += temp
print("Enter Customer Name:", end="")
temp = input()
temp += '%'
data += temp
print("Enter Driver ID:", end="")
temp = input()
temp += '%'
data += temp
print("Enter Type(Instation or Outstation):", end="")
temp = input()
temp += '%'
data += temp
print("Enter Distance(in km):", end="")
temp = input()
temp += '%'
data += temp
print("Enter Fare per km:", end="")
temp = input()
temp += '%'
data += temp
print("Enter Base Fare:", end="")
temp = input()
data += temp

elif (Input == "UBF"):
    print('Enter the date:', end="")
    temp = input()
    temp += '%'
    data += temp
    print('Enter the new Base Fare :', end="")
    temp = input()
    data += temp

elif (Input == "UF"):
    print('Enter the date:', end="")
    temp = input()
    temp += '%'
    data += temp
    print('Enter the new Fare per km :', end="")
    temp = input()
    data += temp

elif (Input == "FD"):
    print('Enter the date:', end="")

```



```

temp = input()
data += temp

elif (Input == "FDr"):
    print('Enter the Driver ID:', end="")
    temp = input()
    data += temp

ClientMultiSocket.send(str.encode(data))
res = ClientMultiSocket.recv(1024)
if Input == "I":
    print(res.decode('utf-8'))
elif Input == "UBF":
    print(res.decode('utf-8'))
elif Input == "UF":
    print(res.decode('utf-8'))
elif Input == "FD":
    x = res.decode('utf-8').split('*')
    fin = {}
    for i in range(1, len(x)):
        if (i % 2 != 0):
            temp = x[i].split("%")
            fin[x[i - 1]] = temp
    # print(fin)
    output = pd.DataFrame.from_dict(fin)
    size = len(output)
    print(output.head(size - 1))

elif Input == "FDr":
    x = res.decode('utf-8').split('*')
    fin = {}
    for i in range(1, len(x)):
        if (i % 2 != 0):
            temp = x[i].split("%")
            fin[x[i - 1]] = temp
    # print(fin)
    output = pd.DataFrame.from_dict(fin)
    size = len(output)
    print(output.head(size - 1))

elif Input == "TC":
    print(res.decode('utf-8'))

elif Input == "T":
    x = res.decode('utf-8').split("%")
    print(x[0], end=":")
    print(x[1])
    print(x[2], end=":")

```

```

        print(x[3])

    elif Input == "V":
        x = res.decode('utf-8').split('*')

        fin = {}
        for i in range(1, len(x)):
            if (i % 2 != 0):
                temp = x[i].split("%")
                fin[x[i - 1]] = temp

        output = pd.DataFrame.from_dict(fin)
        size = len(output)
        print(output.head(size - 1))

ClientMultiSocket.close()

```

Outputs

Initial server and clients' status (Single Server Multi Client):

```

Command Prompt - python server.py
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\KAUSHIK>cd desktop

C:\Users\KAUSHIK\Desktop>python server.py
Socket is listening..
Connected to: 127.0.0.1:60627
Thread Number: 1
Connected to: 127.0.0.1:64299
Thread Number: 2
Connected to: 127.0.0.1:64311
Thread Number: 3

```

```

Command Prompt - python client.py
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\KAUSHIK>cd desktop

C:\Users\KAUSHIK\Desktop>python client.py
Waiting for connection response
I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options:

```

```

Command Prompt - python client.py
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\KAUSHIK>cd desktop

C:\Users\KAUSHIK\Desktop>python client.py
Waiting for connection response
I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options:

```

```

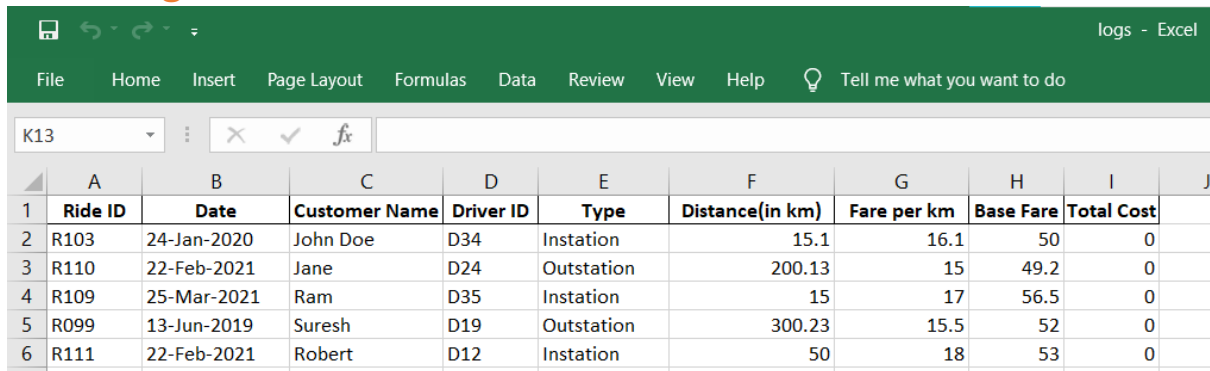
Command Prompt - python client.py
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\KAUSHIK>cd desktop

C:\Users\KAUSHIK\Desktop>python client.py
Waiting for connection response
I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options:

```

Initial logs.xlsx

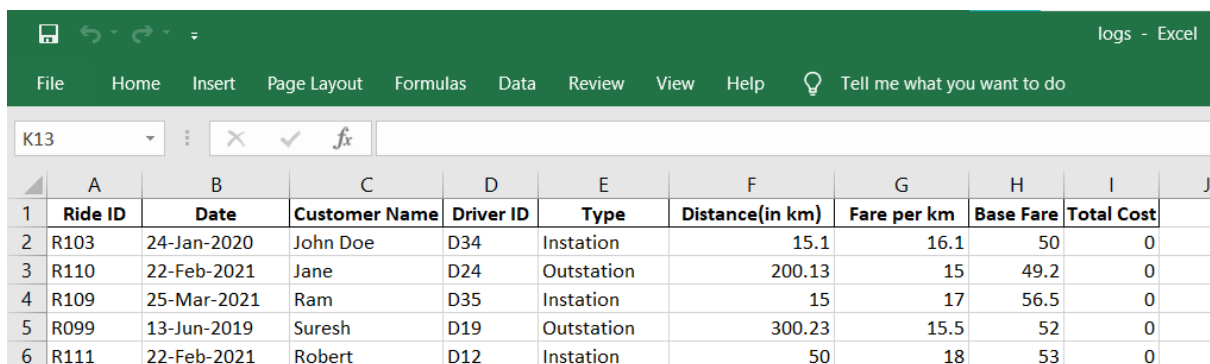


The screenshot shows the Excel interface with the 'logs' workbook open. The data is organized in a table with the following columns: Ride ID, Date, Customer Name, Driver ID, Type, Distance(in km), Fare per km, Base Fare, and Total Cost. The data rows are as follows:

	A	B	C	D	E	F	G	H	I	J
1	Ride ID	Date	Customer Name	Driver ID	Type	Distance(in km)	Fare per km	Base Fare	Total Cost	
2	R103	24-Jan-2020	John Doe	D34	Instation	15.1	16.1	50	0	
3	R110	22-Feb-2021	Jane	D24	Outstation	200.13	15	49.2	0	
4	R109	25-Mar-2021	Ram	D35	Instation	15	17	56.5	0	
5	R099	13-Jun-2019	Suresh	D19	Outstation	300.23	15.5	52	0	
6	R111	22-Feb-2021	Robert	D12	Instation	50	18	53	0	

1.Insert New Record:

```
C:\Users\KAUSHIK\Desktop>python client.py
Waiting for connection response
I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options: I
Enter Ride ID:R123
Enter Date:22-Feb-2021
Enter Customer Name:New
Enter Driver ID:D25
Enter Type(Instation or Outstation):Instation
Enter Distance(in km):30
Enter Fare per km:18
Enter Base Fare:50
Successfully Inserted
```



The screenshot shows the Excel interface with the 'logs' workbook open. The data is organized in a table with the following columns: Ride ID, Date, Customer Name, Driver ID, Type, Distance(in km), Fare per km, Base Fare, and Total Cost. The data rows are as follows:

	A	B	C	D	E	F	G	H	I	J
1	Ride ID	Date	Customer Name	Driver ID	Type	Distance(in km)	Fare per km	Base Fare	Total Cost	
2	R103	24-Jan-2020	John Doe	D34	Instation	15.1	16.1	50	0	
3	R110	22-Feb-2021	Jane	D24	Outstation	200.13	15	49.2	0	
4	R109	25-Mar-2021	Ram	D35	Instation	15	17	56.5	0	
5	R099	13-Jun-2019	Suresh	D19	Outstation	300.23	15.5	52	0	
6	R111	22-Feb-2021	Robert	D12	Instation	50	18	53	0	

3. Update Fare per km based on Date

```

I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options: UF
Enter the date:22-Feb-2021
Enter the new Fare per km :20
Successfully Updated

```

	A	B	C	D	E	F	G	H	I
1	Ride ID	Date	stomer Na	Driver ID	Type	stance(in k	fare per km	Base Fare	Total Cost
2	R103	24-Jan-2020	John Doe	D34	Instation	15.1	16.1	50	0
3	R110	22-Feb-2021	Jane	D24	Outstation	200.13	15	100	0
4	R109	25-Mar-2021	Ram	D35	Instation	15	17	56.5	0
5	R099	13-Jun-2019	Suresh	D19	Outstation	300.23	15.5	52	0
6	R111	22-Feb-2021	Robert	D12	Instation	50	18	100	0
7	R123	22-Feb-2021	New	D25	Instation	30	18	100	0
8									

	A	B	C	D	E	F	G	H	I
1	Ride ID	Date	stomer Na	Driver ID	Type	stance(in k	fare per km	Base Fare	Total Cost
2	R103	24-Jan-2020	John Doe	D34	Instation	15.1	16.1	50	0
3	R110	22-Feb-2021	Jane	D24	Outstation	200.13	20	100	0
4	R109	25-Mar-2021	Ram	D35	Instation	15	17	56.5	0
5	R099	13-Jun-2020	Suresh	D19	Outstation	300.23	15.5	52	0
6	R111	22-Feb-2021	Robert	D12	Instation	50	20	100	0
7	R123	22-Feb-2021	New	D25	Instation	30	20	100	0
8									

4. Fetch all records based on Date

```

I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options: FD
Enter the date:22-Feb-2021

```

	Ride ID	Date	Customer Name	Driver ID	Type	Distance(in km)	Fare per km	Base Fare	Total Cost
0	R110	22-Feb-2021	Jane	D24	Outstation	200.13	20.0	100.0	0
1	R111	22-Feb-2021	Robert	D12	Instation	50.0	20.0	100.0	0
2	R123	22-Feb-2021	New	D25	Instation	30.0	20.0	100.0	0

```

I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options: FDr
Enter the Driver ID:D35

```

Ride ID	Date	Customer Name	Driver ID	Type	Distance(in km)	Fare per km	Base Fare	Total Cost
0 R109	25-Mar-2021	Ram	D35	Instation	15.0	17.0	56.5	0

```
I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options: TC
Successfully Updated
```

[illegible]

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

A1

	A	B	C	D	E	F	G	H	I
1	Ride ID	Date	Customer Name	Driver ID	Type	Distance (in km)	Fare per km	Base Fare	Total Cost
2	R103	24-Jan-20	John Doe	D34	Instation	15.1	16.1	50	293.11
3	R110	22-Feb-20	Jane	D24	Outstation	200.13	20	100	4102.6
4	R109	25-Mar-20	Ram	D35	Instation	15	17	56.5	311.5
5	R099	13-Jun-20	Suresh	D19	Outstation	300.23	15.5	52	4705.565
6	R111	22-Feb-20	Robert	D12	Instation	50	20	100	1100
7	R123	22-Feb-20	New	D25	Instation	30	20	100	700

7. Print no.of each type of rides

```
I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options: T
Instation:4
Outstation:2
```

8. View all

```
I - Insert New Record
UBF - Update Base Fare based on Date
UF - Update Fare per km based on Date
FD - Fetch all records based on Date
FDr - Fetch all records based on Driver ID
TC - Update Total Cost = Base fare+(fare per km*dist)
T - Print no.of each type of rides
V - View all
Choose from the options: V
```

	Ride ID	Date	Customer Name	Driver ID	...	Distance(in km)	Fare per km	Base Fare	Total Cost
0	R103	24-Jan-2020	John Doe	D34	...	15.1	16.1	50.0	293.11
1	R110	22-Feb-2021	Jane	D24	...	200.13	20.0	100.0	4102.6
2	R109	25-Mar-2021	Ram	D35	...	15.0	17.0	56.5	311.5
3	R099	13-Jun-2019	Suresh	D19	...	300.23	15.5	52.0	4705.565000000001
4	R111	22-Feb-2021	Robert	D12	...	50.0	20.0	100.0	1100.0
5	R123	22-Feb-2021	New	D25	...	30.0	20.0	100.0	700.0

[6 rows x 9 columns]

CISCO packet tracer Network design

Carpooling - Subnetting

Human Resource Management	Number of nodes: 4 Range of IP address: 192.168.0.1 – 192.168.0.6 Subnet address: 255.255.255.248 Subnet mask: /29 Protocols configured: FTP, SMTP, HTTP, DNS	purpose of this subnet is that it manages the communication between the customer and the driver. Since there is only HR for any company, the range is very less.
Finance	Number of nodes: 4 Range of IP address: 192.168.2.1- 192.168.2.126 Subnet address:255.255.255.128 Subnet mask:/25 Protocols configured: FTP, SMTP, DNS, HTTP	purpose of this subnet is that it manages all the bill transactions from and to the customer and driver. It requires slightly high number of hosts as there will be many customers and drivers at a single time and that needs to be managed at a single time.
Analytics	Number of nodes: 4 Range of IP address: 192.168.3.1- 192.168.3.126 Subnet address:255.255.255.128 Subnet mask: /25 Protocols configured: HTTP, DNS, FTP, SMTP	purpose of this is that it looks after the security of the payments and the requests that the customers and drivers send and receive. It will be checked for the security of each request posed. Since it is the matter of security, we personally chose the number of hosts to be higher than that of finance dept.
Customer Relations	Number of nodes: 4 Range of IP address: 192.168.4.1- 192.168.4.254 Subnet address:255.255.255.0 Subnet mask: /24 Protocols configured: HTTP, DNS, FTP, SMTP	purpose of this is more or less similar to that of call centre, where customers call representatives to clarify about a query. Since these representatives are needed more with respect to customers, we take the highest number of hosts in this subnet.

VLSM:

Name	Hosts Needed	Hosts Available	Unused Hosts	Network Address	Slash	Mask	Usable Range	Broadcast
Customer Relations	250	254	4	192.168.0.0	/24	255.255.255.0	192.168.0.1 - 192.168.0.254	192.168.0.255
Analytics	120	126	6	192.168.1.0	/25	255.255.255.128	192.168.1.1 - 192.168.1.126	192.168.1.127
Finance	70	126	56	192.168.1.128	/25	255.255.255.128	192.168.1.129 - 192.168.1.254	192.168.1.255
Human Resources	3	6	3	192.168.2.0	/29	255.255.255.248	192.168.2.1 - 192.168.2.6	192.168.2.7
Main Server	2	2	0	192.168.2.8	/30	255.255.255.252	192.168.2.9 - 192.168.2.10	192.168.2.11

- Routers follow class A ip addresses.
 - Remaining devices follow class C.
 - We are not using class B as the network address is hard to follow up with while directing and redirecting the network.
-
- The address block for main server – 192.168.0.0/24 – number of hosts only 2
 - The address block for HR server—192.168.1.0/24 – number of hosts only 3
 - The address block for Finance server – 192.168.2.0/24 – number of hosts 70
 - The address block for Analytics server – 192.168.3.0/24 – number of hosts 120
 - The address block for Customer relations server – 192.168.4.0/24 – number of hosts 250

Why not continuous subnetting? Why change the 2nd octet for each subnetwork?

This is an example and the scale of this network is very small. We do not certainly know when new employees get their jobs and in which division.

So, to maintain clarity and no overlapping of ip addresses, we gave a new 2nd octet to each subnet.

NODES:

HR management:

Nodes:- main server router, switch in main server router, HR router, switch in hr router—4

Finance:

Nodes:- main server router, switch in main server router, Finance router, switch in finance router—4

Analytics:

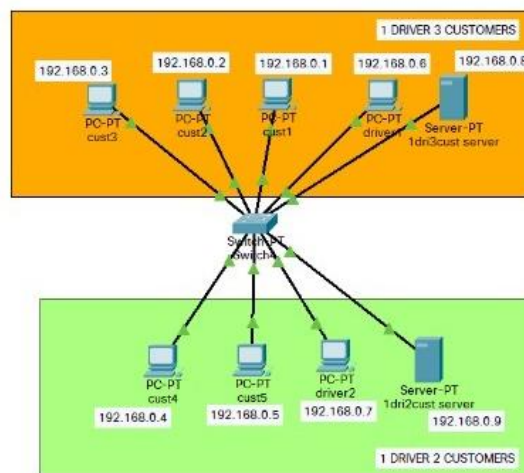
Nodes:- main server router, switch in main server router, Analytics router, switch in analytics router—4

Customer Relations:

main server router, switch in main server router, Customer relations router, switch in customer relations router—4

VLAN

While using vlan, only the customer and driver can communicate among themselves leaving the admin in vain. The admin will not have any chance of knowing what the customer and driver are conversing. This is a drawback if vlan is used. But the main goal of this network can be achieved only with the help of VLAN.



HR MANAGEMENT SUBNET:

SERVER NAME: HR Management

Protocols:

SMTP—it is a mail transfer protocol which is used to communicate between every subnet and host except analytics subnet.

FTP – it is a file transfer protocol which is used to send and receive files from all the hosts including analytics subnet hosts.

FINANCE SUBNET:

SERVER NAME: Finance

Protocols:

SMTP—it is a mail transfer protocol which is used to communicate between every subnet and host except analytics subnet.

FTP – it is a file transfer protocol which is used to send and receive files from all the hosts including analytics subnet hosts.

CUSTOMER RELATIONS SUBNET:

SERVER NAME: Customer Relations

Protocols:

SMTP—it is a mail transfer protocol which is used to communicate between every subnet and host except analytics subnet.

HTTP – Since it is a customer relation subnet, the major functionality is that they keep in touch with each and every customer. So, HTTP and SMTP is a way to connect to customers.

ANALYTICS SUBNET:

SERVER NAME: Analytics

Protocols:

FTP – it is a file transfer protocol which is used to send and receive files from all the hosts including analytics subnet hosts. It is modified in such a way that the file that get transferred stays in that subnet and does not get out as this subnet takes care of the security of delicate info.

SMTP – this protocol is only available to analytics division employees as confidentiality is very much needed for security.

NETWORK DEVICES: (throughout the server)

- Switch – Since routers do not have many ports to connect multiple devices, we pass the connections through these devices.
- PC – Ultimate device where the hosts access the whole server.
- Server – The device manages the protocols that each host can have access to host.
- Router – the device from which the connections originate and reach the further devices.

Routing Algorithm

Selective Repeat:

CLIENT SIDE:

```
import java.lang.System;
import java.net.*;
import java.io.*;

public class Client {
    static Socket connection;

    public static void main(String a[]) throws SocketException {
        try {
            int v[] = new int[9];
            //int g[] = new int[8];
            int n = 0;
            InetAddress addr = InetAddress.getByName("Localhost");
            System.out.println(addr);
            connection = new Socket(addr, 8011);
            DataOutputStream out = new DataOutputStream(
                connection.getOutputStream());
            DataInputStream in = new DataInputStream(
                connection.getInputStream());
            int p = in.read();
            System.out.println("No of frame is:" + p);

            for (int i = 0; i < p; i++) {
                v[i] = in.read();
                System.out.println(v[i]);
                //g[i] = v[i];
            }
            v[5] = -1;
            for (int i = 0; i < p; i++)
            {
                System.out.println("Received frame is: " + v[i]);

            }
            for (int i = 0; i < p; i++)
            if (v[i] == -1) {
                System.out.println("Request to retransmit packet no "
                    + (i+1) + " again!!");
                n = i;
                out.write(n);
                out.flush();
            }

            System.out.println();

            v[n] = in.read();
            System.out.println("Received frame is: " + v[n]);
```

```

        System.out.println("quiting");
    } catch (Exception e) {
        System.out.println(e);
    }
}
}
}

```

SERVER SIDE:

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;

public class Server {
    static ServerSocket Serversocket;
    static DataInputStream dis;
    static DataOutputStream dos;

    public static void main(String[] args) throws SocketException {

        try {
            int a[] = { 30, 40, 50, 60, 70, 80, 90, 100, 110 };
            Serversocket = new ServerSocket(8011);
            System.out.println("waiting for connection");
            Socket client = Serversocket.accept();
            dis = new DataInputStream(client.getInputStream());
            dos = new DataOutputStream(client.getOutputStream());
            System.out.println("The number of packets sent is:" + a.length);
            int y = a.length;
            dos.write(y);
            dos.flush();

            for (int i = 0; i < a.length; i++) {
                dos.write(a[i]);
                dos.flush();
            }

            int k = dis.read();

            dos.write(a[k]);
            dos.flush();

        } catch (IOException e) {
            System.out.println(e);
        } finally {
            try {
                dis.close();
            }
        }
    }
}

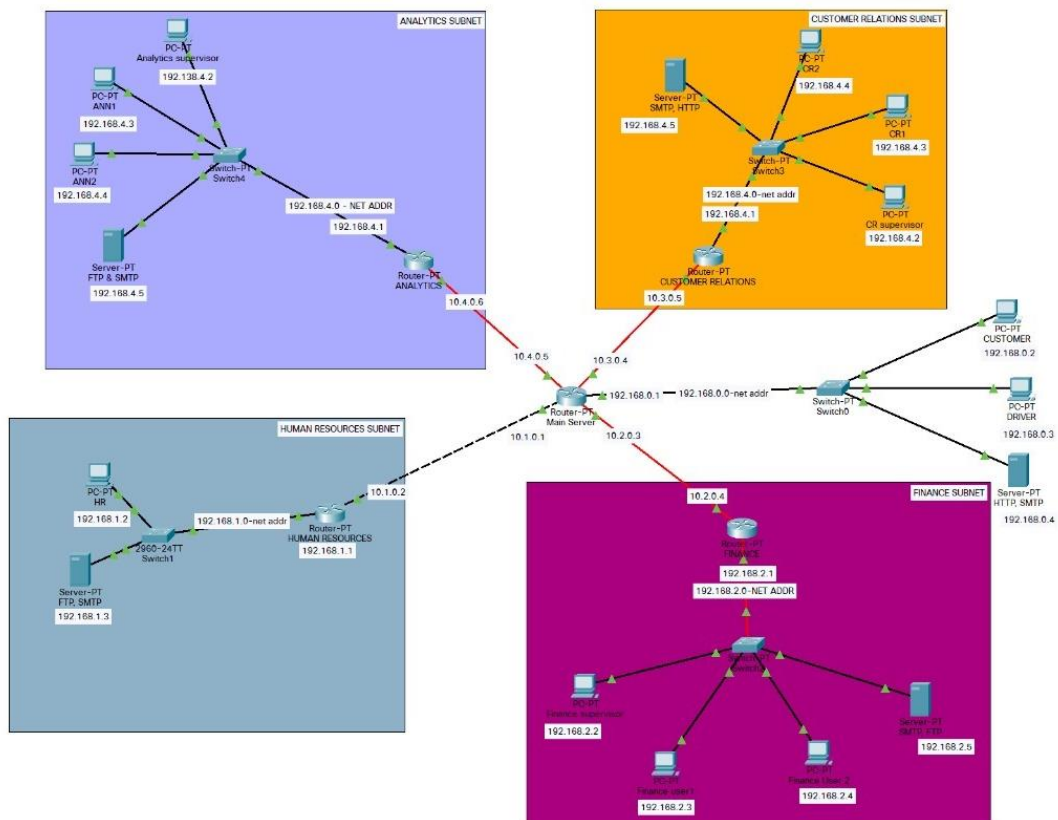
```

```

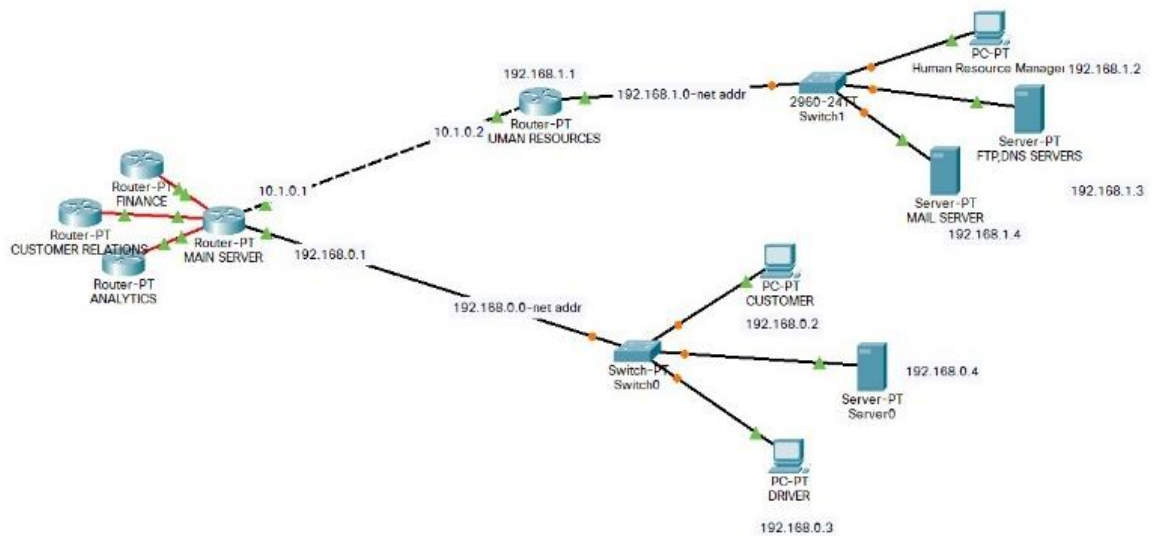
dos.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}
}

```

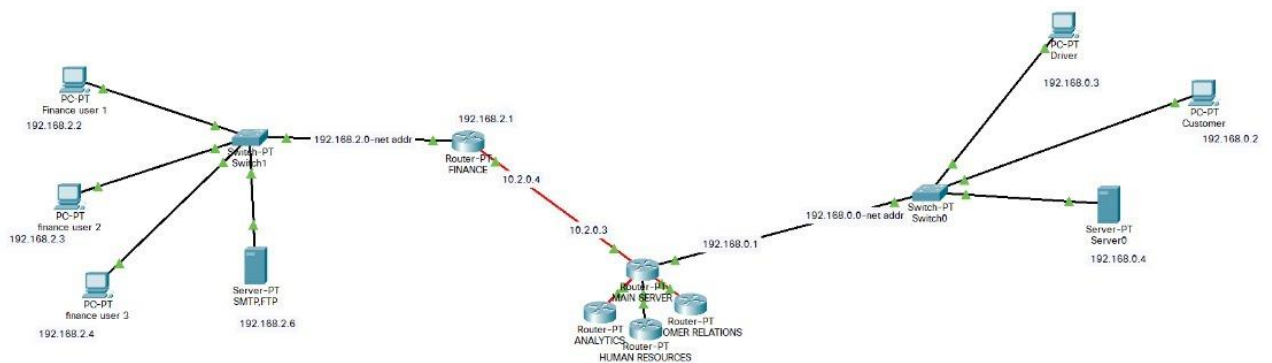
SCREENSHOTS:



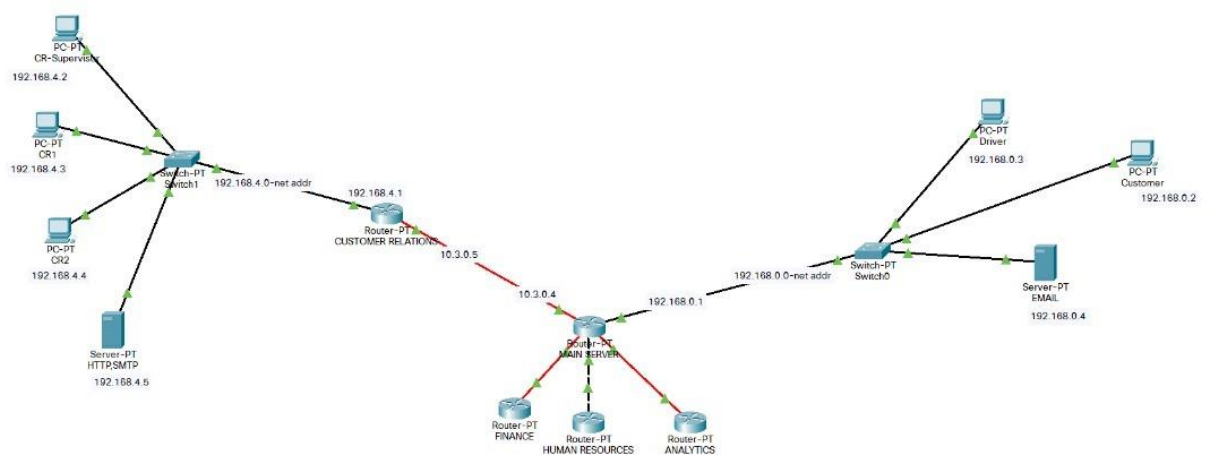
HR subnet:



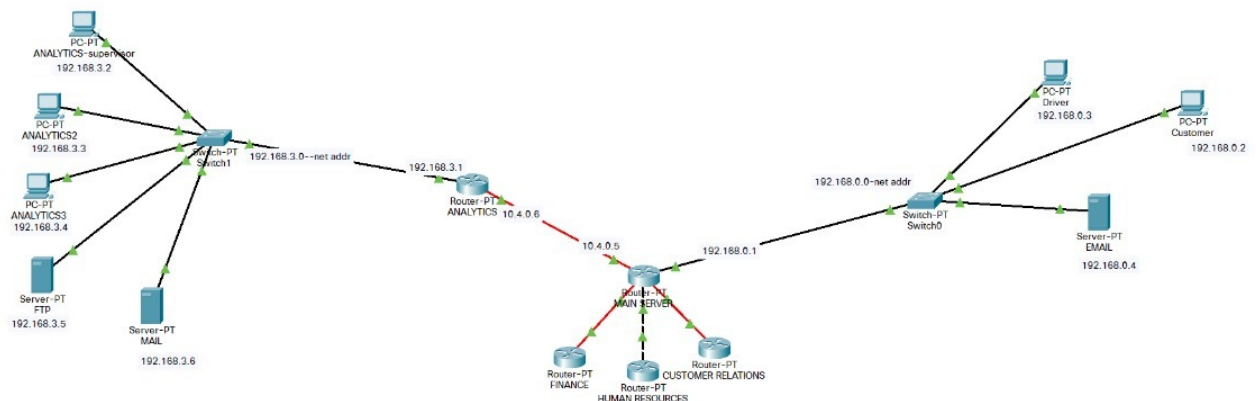
Finance subnet:



Customer relations subnet:



Analytics subnet:



Cloud Computing and Virtualization:

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centres and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider. Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications. For example, healthcare companies are using the cloud to develop more personalized treatments for patients. Financial services companies are using the cloud to power real-time fraud detection and prevention. And video game makers are using the cloud to deliver online games to millions of players around the world.

Virtualization:

This firm uses private mode of cloud virtualization as it's easy for us to aggregate our resources in the data centre into a single pool of resources so that both our employees and clients be benefitted. Through virtualization of the hardware components, our company gets an increase in the efficiency and utilization of our private cloud infrastructure.

1. Storage Virtualization:

Storage virtualization (also sometimes called software-defined storage or a virtual SAN) is the pooling of multiple physical storage arrays from SANs and making them appear as a single virtual storage device. Cloud servers can be used to save data on cloud storages. Having all the data stored in the cloud ensures it is backed up and protected in a secure and safe location. In case of any breach in security or failure of local storage drives, cloud drives will maintain backups across multiple storage drives.

2. Server Virtualization:

Virtualization in Cloud Computing is a process in which the user of the cloud shares the data present in the cloud which can be application software etc. It provides a virtual environment in the cloud which can be software hardware or any other thing. Server Virtualization here is used to offload heavy and intensive tasks such as OS-level virtualization using software's like Docker, heavy builds onto high computation private servers used in the firm. These tasks are resource intensive and normal computers fail to run these as they lack the memory and computation power to run these tasks. A

segment of the server, or a box, can be requested in one of the servers which can be used to do these heavy tasks while the server itself is not fully used by a single client.

Conclusion:

The packet file that was shown is just reduced to a programmable scale. There can be any number of hosts as well as subnets. We have implemented all the necessary basic and routing protocols that are majorly necessary. The basic conditions that are required in our case study have been achieved. Cloud Virtualisation has been described in detailed as well as Selective repeat algorithm with respect to our case study. Socket programming has been done in Python language along with multi-threading. In this network, the basic condition, that is no two drivers can communicate among themselves, has been shown with the help of VLAN.