15CSE312- COMPUTER NETWORK

HOSPITAL INFORMATION SYSTEM(HIS)

Group Number 17

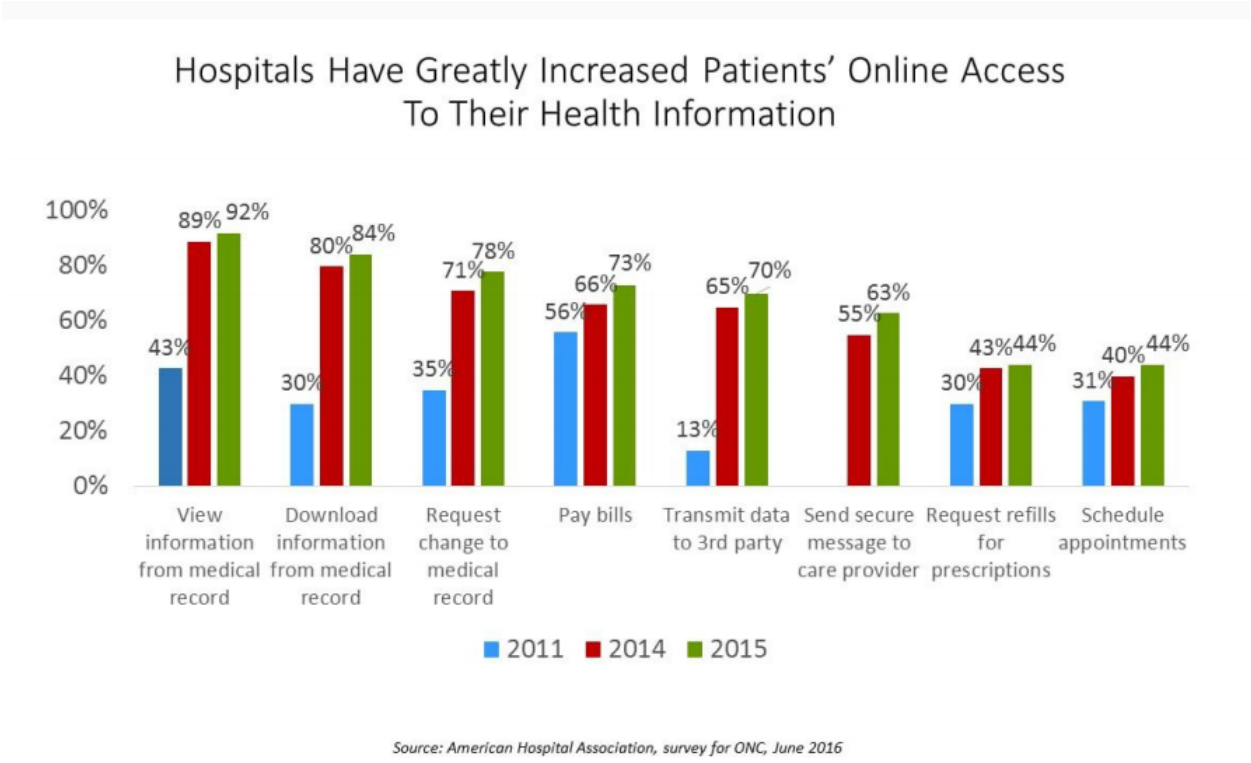
Registration No	Name	Email ID	IndividualContribution
CB.EN.U4CSE18011	B.Sai Madhav	cb.en.u4cse18011@cb.students.amrita.edu	Cisco Packet tracer,Analytical Question, Architecture diagram , Abstract
CB.EN.U4CSE18017	G.Rohith Guptha	cb.en.u4cse18017@cb.students.amrita.edu	Cisco Packet tracer,socket programming
CB.EN.U4CSE18032	K.Anudeep	cb.en.u4cse18032@cb.students.amrita.edu	Routing algorithm, Comparison of routing algorithms,Cisco Packet tracer

Why Networking is required for the application:

Critical medical data must be accessible exactly where and when they're needed, yet many systems are currently struggling to make their data accessible in ways that match the real-time contexts of today's healthcare.

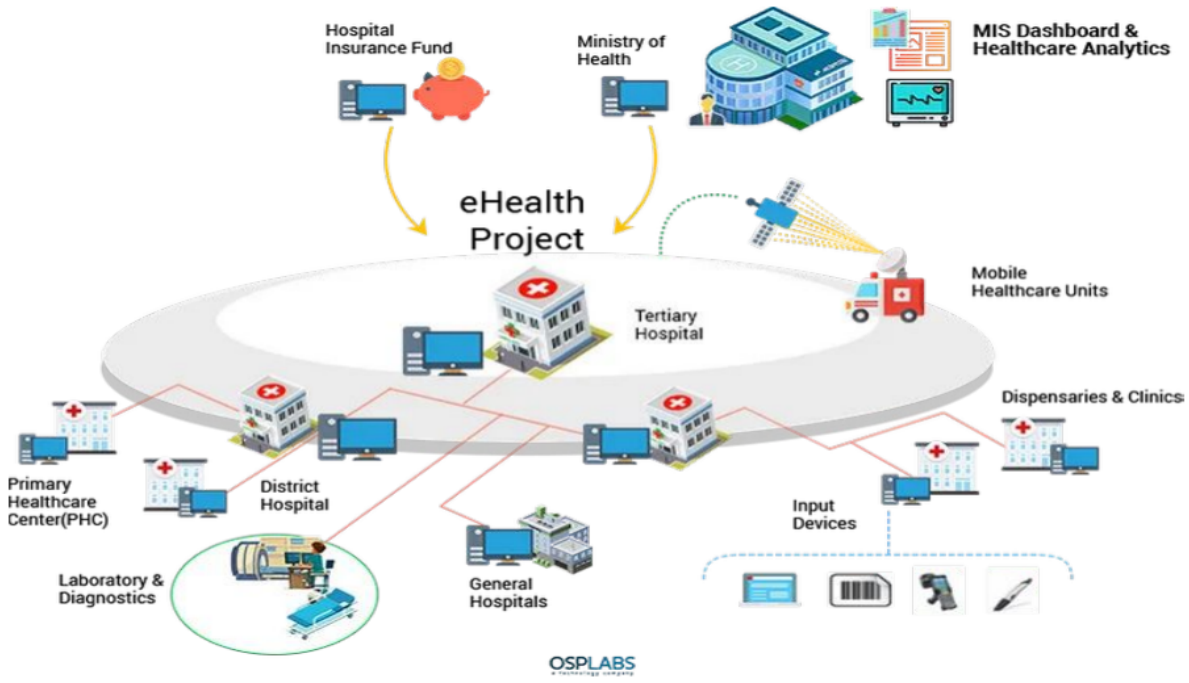
Multiple branches of the same hospital might be connected in the same network. Various doctors in one network can access a patient's record.

Patient in this network can get a report of his visit The need for security and encryption to protect the confidentiality of patient data



Source: American Hospital Association, survey for ONC, June 2016

Case study on Hospital information management system



Problem statement:

To study the network structure in an Hospital Network (Chain of Hospitals i.e., Apollo Hospitals).To understand its topology and understand its Architectural design.Also,to analyze the network by simulating using cisco packet tracer.Also,trying and comparing various routing algorithms.

Abstract :

A hospital information system (HIS) is an element of health informatics that focuses mainly on the administrational needs of hospitals. In many implementations, a HIS is a comprehensive, integrated information system designed to manage all the aspects of a hospital's operation, such as medical, administrative, financial, and legal issues and the corresponding processing of services. A hospital information system is also known as hospital management software (HMS) or hospital management system. Hospital information systems provide a common source of information about a patient's health history. The system has to keep data in a secure place and control who can reach the data in certain circumstances. These systems enhance the ability of health care

professionals to coordinate care by providing a patient's health information and visit history at the place and time that it is needed. The patient's laboratory test information also includes visual results such as X-ray, which may be reachable by professionals. HIS provides internal and external communication among health care providers. Portable devices such as smartphones and tablet computers may be used at the bedside. Hospital information systems are often composed of one or several software components with specialty-specific extensions, as well as of a large variety of sub-systems in medical specialties.

How Networks Work in Hospital Information Management?

The following Step explains how networks work in Hospital Management

- Patient enters his/her details
- Data entered goes through cloud server into Database.
- Doctor/Patient in our Network is able to access the data from the database

Benefits of computer networks in Hospital Information Management:

- Improved monitoring of drug usage, and study of effectiveness. This leads to the reduction of adverse drug interactions while promoting more appropriate pharmaceutical utilization
- Efficient and accurate administration of finance, diet of patient, engineering, and distribution of medical aid. It helps to view a broad picture of hospital growth
- Enhances information integrity, reduces transcription errors, and reduces duplication of information entries

Analytical Questions :

- Which of the following describes the creation of private networks across the Internet, enabling privacy and tunneling of non-TCP/IP protocols?
- How does a network topology affect your decision to set a network?
- What is the importance of implementing a Fault Tolerance System?

- What is the disadvantage of a star topology?
- When it comes to networking, what are rights?
- How does one prevent bottlenecks in the network, i.e., manage flow control?
- Quantify the scalability of the network - i.e, when more users connect, does the network suffer?

Protocols in Hospital Information Management:

- Tcp
- Internet protocol

Data Tables :

Parameters	Specifications
Number of Nodes	N Branches = n Nodes
Type of Addressing	Classless Addressing.
Network Devices	Routers,Switches.
Routing Protocol	Probabilistic Shortest Path Routing Algorithm.

Communication Media	Optical Fibre Cable
IP Address Range	198.168.10.1 to 198.168.10.109
Type of Network	WAN

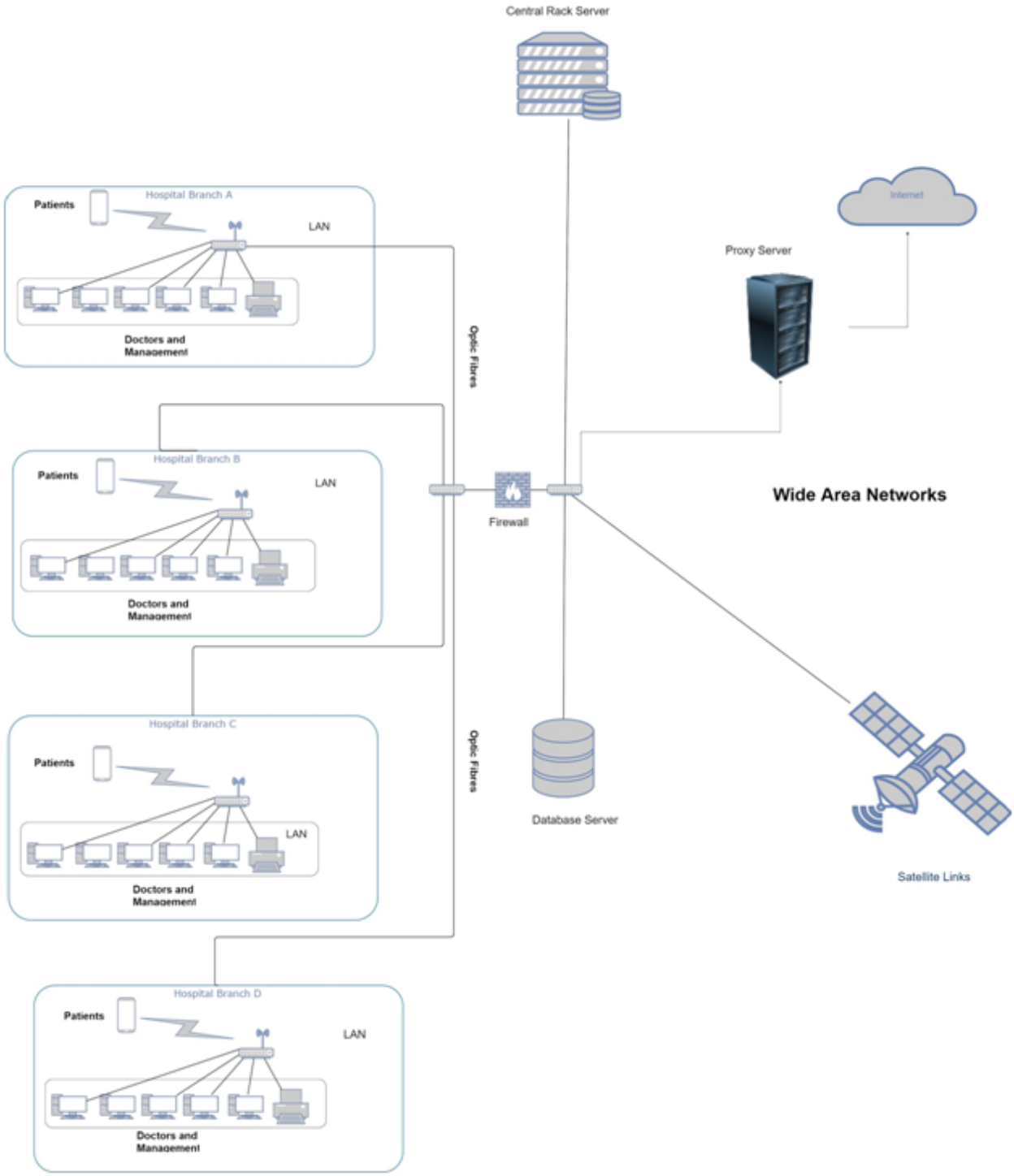
Performance parameters:

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
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.	Directly proportional to processing speed of the routers.
Queuing Delay	Time spent by the data packet waiting in the queue before it is taken for execution is called queuing delay.	Directly Proportional to the congestion in the network

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).

Architecture diagram:



Socket Program:

HOSPITAL INFORMATION SYSTEM:

In this scenario we will use *multi client* and *single server* socket programming techniques.

On the server side an excel sheet named DATABASE_HOSPITAL is created with the following parameters:

1. patient_id
2. p_name
3. doctor_id
4. d_name
5. date
6. time
7. report
8. tests
9. perception

After the above file has been created on the server side the clients will send the inputs which must be read by the server.

Code:

Server Side:-

```
import socket

import threading

import pandas

import json
```

```
HEADER = 64

PORT = 5050

SERVER = socket.gethostbyname(socket.gethostname())

ADDR = (SERVER, PORT)

FORMAT = 'utf-8'

DISCONNECT_MESSAGE = "!DISCONNECT"


server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server.bind(ADDR)



def server_function():

   df = pandas.read_csv('lab3.csv')

   patient_id = input("Enter the patient_id")

   for i in range(len(df)):

      if(df.iloc[i, 0] == patient_id):

         print(df.iloc[i])

         print("\n")



def handle_client(conn, addr):
```

```
    print(f"[NEW CONNECTION] {addr} connected.")
    connected = True
    while connected:


        msg_length = conn.recv(HEADER).decode(FORMAT)
        if msg_length:


            df = pandas.read_csv('lab3.csv')


            msg_length = int(msg_length)
            msg = conn.recv(msg_length).decode(FORMAT)
            if msg == DISCONNECT_MESSAGE:
                connected = False
            userinput = json.loads(msg)
            print(userinput)
            print(f"[{addr}] {userinput}")


            df2 = {'patient_id': userinput['patient_id'], 'p_name':
userinput['p_name'],'doctor_id': userinput['doctor_id'], 'd_name': userinput['d_name'],
                   'date': userinput['date'], 'time': userinput['time'], 'report': userinput['report'],
'tests': userinput['tests'], 'perception': userinput['perception']}
            df = df.append(df2, ignore_index=True)
            df.to_csv('lab3.csv', index=False)
            print(df)
```

```
            conn.send("Values are Updated".encode(FORMAT))

    conn.close()



def start():

    server.listen()
    print(f"[LISTENING] Server is listening on {SERVER}")

    x = int(input("press 1- server function"))
    if x == 1:
        server_function()

    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")



print("[STARTING] server is starting...")
```

```
start()
```

Client Side:-

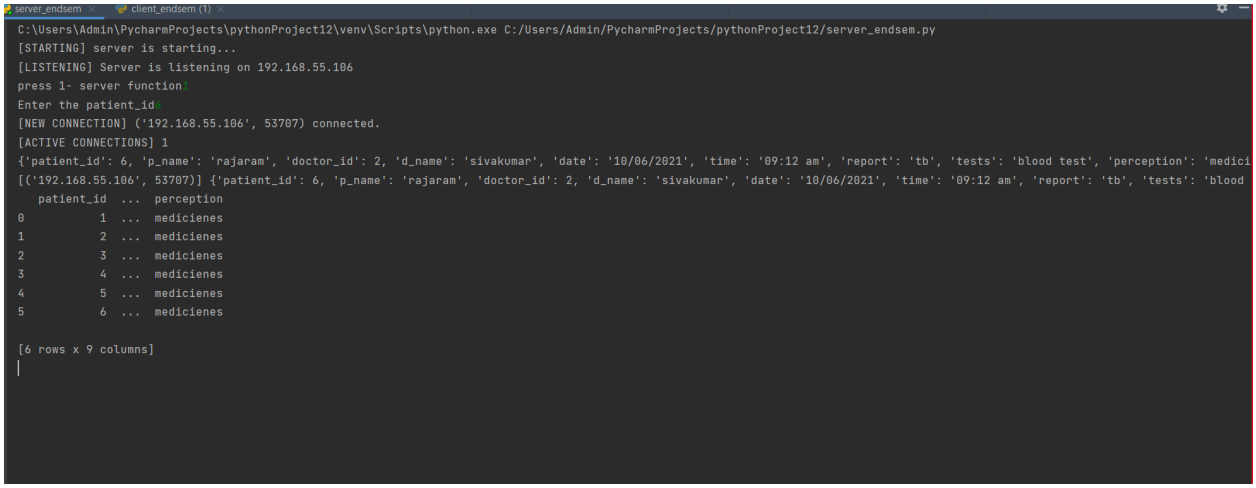
```
import socket
import json

HEADER = 64
PORT = 5050
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)


def send(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
```

```
    client.send(send_length)
    client.send(message)
    print(client.recv(2048).decode(FORMAT))
```

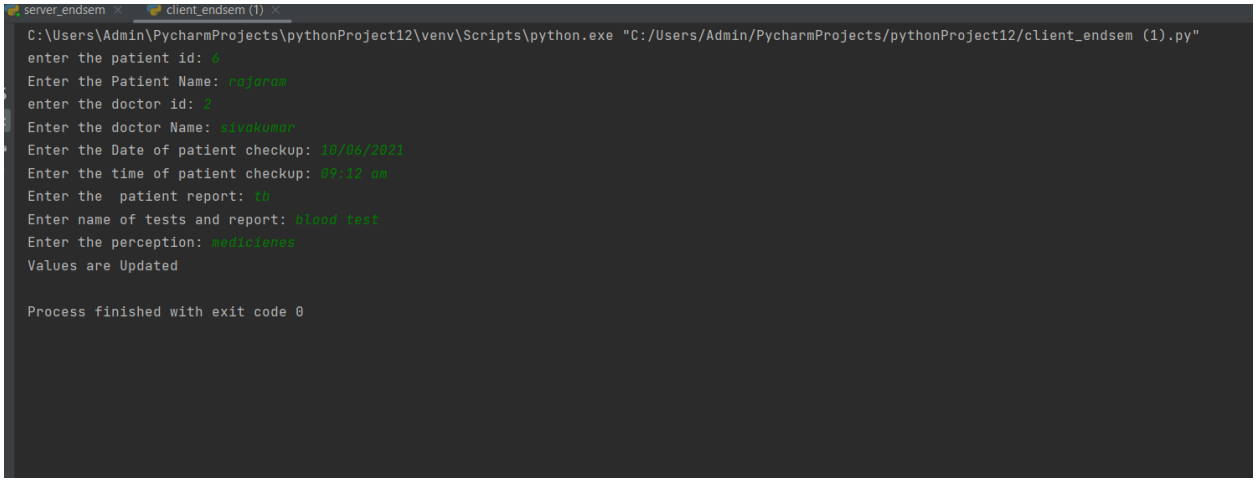
```
patient_id =int(input("enter the patient id: "))
p_name = input("Enter the Patient Name: ")
doctor_id=int(input("enter the doctor id: "))
d_name = input("Enter the doctor Name: ")
date = input("Enter the Date of patient checkup: ")
time = input("Enter the time of patient checkup: ")
report = input("Enter the  patient report: ")
tests = input("Enter name of tests and report: ")
perception= input("Enter the perception: ")

x = {'patient_id': patient_id, 'p_name': p_name,  'doctor_id': doctor_id, 'd_name':
d_name,'date': date, 'time': time, 'report': report, 'tests': tests,'perception':perception}
y = json.dumps(x)
send(y)
```

Output:

Server Side:-



Client Side:-



Excel Sheet:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	patient_id	p_name	doctor_id	d_name	date	time	report	tests	perception				
2	1	rakesh	1	gopal	########	09:00 AM	suffering f	mri scanni	medicienes				
3	2	siva	2	rajesh	########	09:00 AM	suffering f	mri scanni	medicienes				
4	3	chiru	1	rohith	########	09:00 AM	suffering f	mri scanni	medicienes				
5	4	madhav	2	sriram	########	09:00 AM	suffering f	mri scanni	medicienes				
6	5	harish	3	tarun	########	09:00 AM	suffering f	mri scanni	medicienes				
7	6	rajaram	2	sivakumar	########	09:12 AM	tb	blood test	medicienes				
8													
9													
10													
11													

Routing Algorithm :

Here we used dijkstra's algorithm to find the shortest path between the routers. Multiconstrained optimization methods for path selection, with more than one metric to be optimized, are highly challenging and have been proved to be NP-complete. A problem is NP-complete if all decisions can be verified in polynomial time.To achieve single-objective optimization for routing, Dijkstra's algorithm can be applied to a network. This algorithm finds the shortest path between two nodes by developing the path in order of increasing path length. It has the advantage of fast shortest path determination and a computational complexity of $O(n^2)$, where n is the number of routers, rendering it efficient for use with relatively large networks.

1) So this algorithm will identify all of the vertices that are connected to the current vertex with an edge. Calculate their distance to the end by adding the weight of the edge to the mark on the current vertex. Mark each of the vertices with their corresponding distance, but only change a vertex's mark if it's less than a previous mark. Each time you mark the starting vertex with a mark, keep track of the path that resulted in that mark.

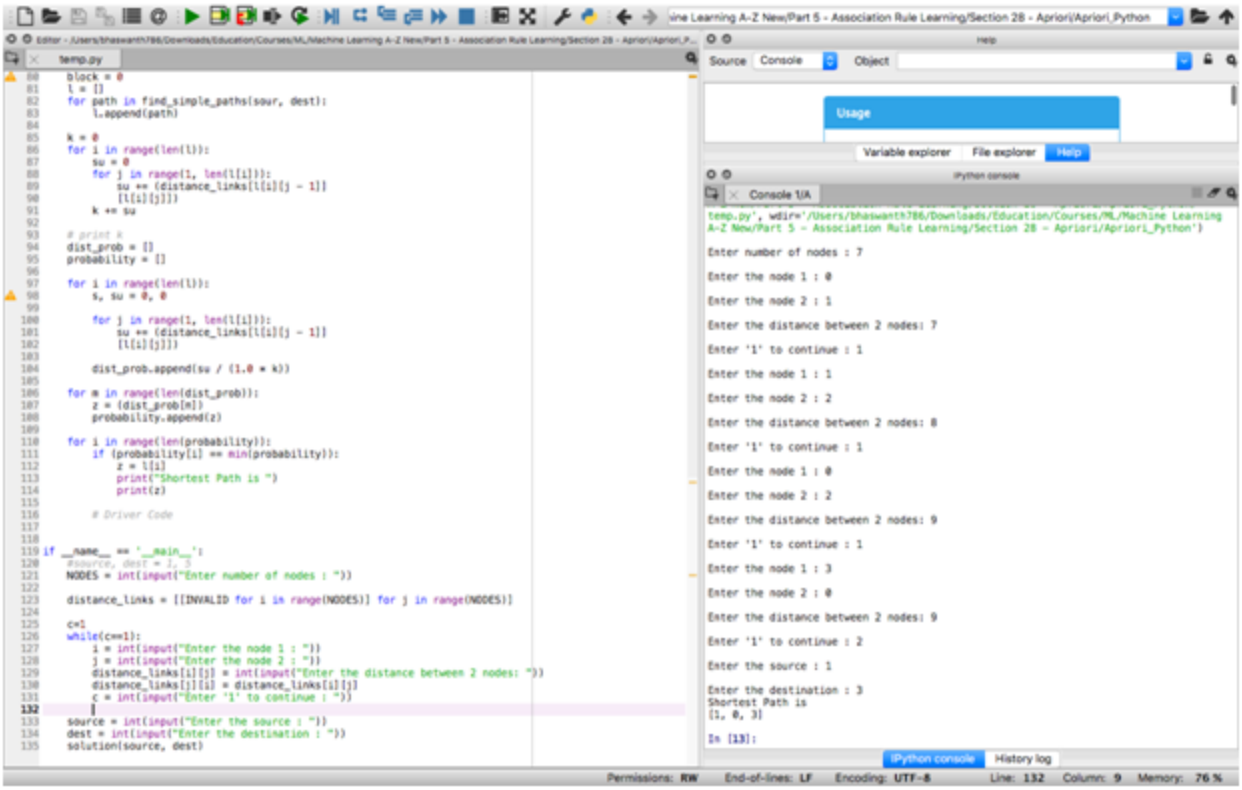
2) Label the current tot vertex as visited by putting an X over it. Once a vertex is visited, we won't look at it again.

3) Of the vertices you just marked, find the one with the smallest mark, and make it your current vertex. Now, you can start again from step 1.

4) Once you've labeled the beginning vertex as visited - stop. The distance of the shortest path is the mark of the starting vertex, and the shortest path is the path that resulted in that mark.

5) Let's now consider finding the shortest path for the 2 desired routers and it will find the optimized shortest path.

Screenshot of the implementation :



Python code to implement routing algorithm:

```
import random


#NODES = 7


INVALID = 0.001


def next_node(s):
      nxt = []


      for i in range(NODES):
      if (distance_links[s][i] != INVALID):
            nxt.append(i)
```

```
      return nxt



def find_simple_paths(start, end):
      visited = set()
      visited.add(start)

      nodestack = list()
      indexstack = list()
      current = start
      i = 0

      while True:

      # get a list of the neighbors
      # of the current node
      neighbors = next_node(current)

      # Find the next unvisited neighbor
      # of this node, if any
      while i < len(neighbors) and neighbors[i] in visited:
            i += 1

      if i >= len(neighbors):
            visited.remove(current)
```

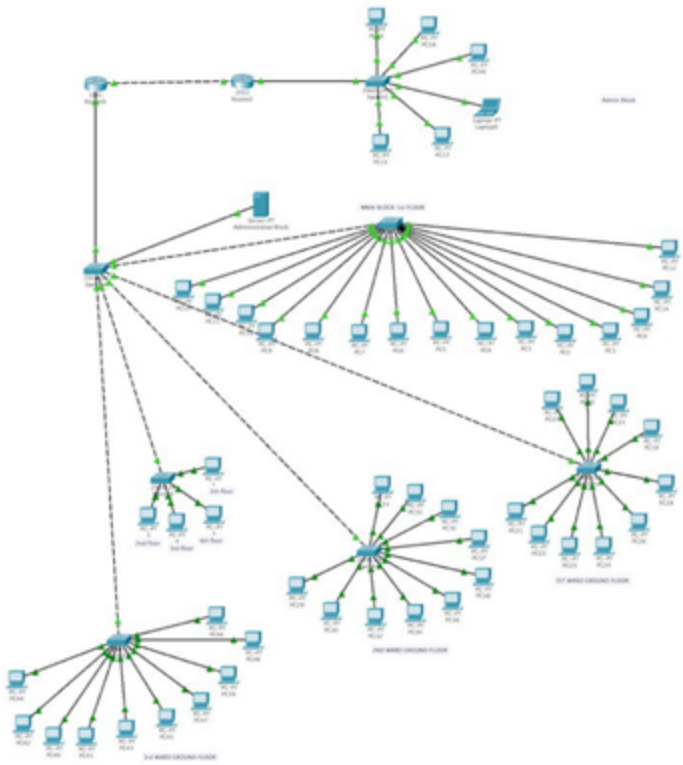
```
            if len(nodestack) < 1:
                  break

            current = nodestack.pop()
            i = indexstack.pop()

      elif neighbors[i] == end:
            yield nodestack + [current, end]
            i += 1

      else:
            nodestack.append(current)
            indexstack.append(i + 1)
            visited.add(neighbors[i])
            current = neighbors[i]
            i = 0
def solution(sour, dest):
      block = 0
      l = []
      for path in find_simple_paths(sour, dest):
      l.append(path)

      k = 0
      for i in range(len(l)):
```

```
    su = 0
    for j in range(1, len(l[i])):
          su += (distance_links[l[i][j - 1]]
          [l[i][j]])
    k += su
    dist_prob = []
    probability = []

    for i in range(len(l)):
    s, su = 0, 0

    for j in range(1, len(l[i])):
          su += (distance_links[l[i][j - 1]]
          [l[i][j]])

    dist_prob.append(su / (1.0 * k))

    for m in range(len(dist_prob)):
    z = (dist_prob[m])
    probability.append(z)

    for i in range(len(probability)):
    if (probability[i] == min(probability)):
          z = l[i]
          print("Shortest Path is ")
```

```
            print(z)


      # Driver Code
if __name__ == '__main__':
      NODES = int(input("Enter number of nodes : "))
      distance_links = [[INVALID for i in range(NODES)] for j in
range(NODES)]


      c=1
      while(c==1):
      i = int(input("Enter the node 1 : "))
      j = int(input("Enter the node 2 : "))
      distance_links[i][j] = int(input("Enter the distance between 2
nodes: "))
      distance_links[j][i] = distance_links[i][j]
      c = int(input("Enter '1' to continue : "))


      source = int(input("Enter the source : "))
      dest = int(input("Enter the destination : "))
      solution(source, dest)
```

Cisco Packet Tracer File:

Cisco packet tracer is a cross platform visual simulation tool designed by cisco systems that allows users to create network topologies and imitate modern computer networks.The software allows users to simulate the configuration of Cisco routers and switches using a simulated command line interface. Packet Tracer makes use of a drag and drop user interface, allowing users to add and remove simulated network devices as they see fit.



Network Structure :

Our application has 6 departments

1.Administrative Block

2.Main Block (1st Floor)

3.1st ward (GF)

4.2nd ward (GF)

5..3rd ward (GF)

6.Floors

Ip addressing Table:

Department	**Subnet Mask**	**/n**	**Network Address**	Broadcast Address	Usable addresses
Administrative Block	255.255.255.248	/29	192.168.10.0	192.168.10.7	6
Main Block (1st Floor)	255.255.255.224	/27	192.168.10.8	192.168.10.39	30
1st ward (GF)	255.255.255.240	/28	192.168.10.40	192.168.10.55	14
2nd ward (GF)	255.255.255.240	/28	192.168.10.56	192.168.10.71	14
3rd ward (GF)	255.255.255.240	/28	192.168.10.72	192.168.10.77	14

2nd Floor	255.255.255.248	/29	192.168.10.78	192.168.10.85	6
3rd Floor	255.255.255.248	/29	192.168.10.86	192.168.10.93	6
4th floor	255.255.255.248	/29	192.168.10.94	192.168.10.101	6
5th floor	255.255.255.248	/29	192.168.10.102	192.168.10.109	6

Reference

https://www.computernetworkingnotes.com/ccna-study-guide/vlsm-subnetting-explained-with-examples.html