LAB FILE

JAMIA MILLIA ISLAMIA

DEPARTMENT OF COMPUTER ENGINEERING

B. TECH. (COMPUTER ENGINEERING)

5th SEMESTER

COMPUTER NETWORKS Lab
(CEN-593)

Submitted To:

Prof. Mohammad Amjad

Prof. Hannan Mansoor

TEAM MEMBER DETAILS:

Sr. No.	NAME	Roll No.
1.	Md. Ibrahim Akhtar	21BCS007
2.	Mohammad Faraz Idris Siddiqui	21BCS009
3.	Mohammad Yasir Aftab	21BCS010
4.	Osama Ayub Ghazi	21BCS023
5.	Md. Masleuddin	21BCS028

Index

LAB 1

AIM: Write a Program to implement Ceaser Cipher.

```
#include <bits/stdc++.h>
using namespace std;

void decrypt(string &text, int key)
{
    cout << "Encrypted String:-  " << text << endl;
    for (int i = 0; i < text.length(); i++)
    {
        if (text[i] >= 'a' && text[i] <= 'z')
        {
            int temp = text[i] - 'a';
            temp = (temp - key) % 26;
            if (temp < 0)
            {
                temp = temp + 26;
            }
            text[i] = temp + 'a';
        }
        else if (text[i] >= 'A' && text[i] <= 'Z')
        {
            int temp = text[i] - 'A';
            temp = (temp - key) % 26;
            if (temp < 0)
            {
                temp = temp + 26;
            }
            text[i] = temp + 'A';
        }
    }
    cout << "Decrypted String:-  " << text << endl;
}
void enycrypt(string &text, int key)
{
    cout << "Given String:-  " << text << endl;
    for (int i = 0; i < text.length(); i++)
    {
        if (text[i] >= 'a' && text[i] <= 'z')
        {
            int temp = text[i] - 'a';
            temp = (temp + key) % 26;
            text[i] = temp + 'a';
        }
        else if (text[i] >= 'A' && text[i] <= 'Z')
        {
```

```
            int temp = text[i] - 'A';
            temp = (temp + key) % 26;
            text[i] = temp + 'A';
        }
    }
    cout << "Encrypted String:-  " << text << endl;
}
int main()
{
    string text;
    cout << "Enter the string:-  ";
    getline(cin, text);
    int key;
    cout << "Enter the key:-  ";
    cin >> key;
    bool flag = false;
    while (1)
    {
        int option;
        cout << "1. Encrypt\n2. Decrypt\n3. Exit\n";
        cin >> option;
        if (option == 1)
        {
            flag = true;
            enycrypt(text, key);
        }
        else if (option == 2)
        {
            if (!flag)
            {
                cout << "Please Encrypt the string first\n";
                continue;
            }
            decrypt(text, key);
            flag = false;
        }
        else if (option == 3)
        {
            break;
        }
        else
        {
            cout << "Invalid Option\n";
        }
    }
    return 0;
}
```

OUTPUT:

```
PS D:\Programming\CompNetworks Lab 5th Sem> g++ -o cc Ceaser_Cipher.cpp
PS D:\Programming\CompNetworks Lab 5th Sem> ./cc
Enter the string:-  Hello World how are you
Enter the key:-  4
1. Encrypt
2. Decrypt
3. Exit
1
Given String:-  Hello World how are you
Encrypted String:-  Lipps Asvph lsa evi csy
1. Encrypt
2. Decrypt
3. Exit
2
Encrypted String:-  Lipps Asvph lsa evi csy
Decrypted String:-  Hello World how are you
1. Encrypt
2. Decrypt
3. Exit
3
PS D:\Programming\CompNetworks Lab 5th Sem>
```

LAB 2

AIM: Write a Program to implement Transposition Cipher.

```cpp
#include <bits/stdc++.h>
using namespace std;

void Enycrypt(string &str, string key)
{
    cout << "Given String:-  " << str << endl;
    int n = str.length();
    int m = key.length();
    int row = n / m;
    if (n % m != 0)
    {
        row++;
    }
    char mat[row][m];
    int k = 0;
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < m; j++)
        {
            if (k < n)
```

```
            {
                mat[i][j] = str[k++];
            }
            else
            {
                mat[i][j] = ' ';
            }
        }
    }
    string temp = key;
    sort(temp.begin(), temp.end());
    str.clear();
    for (int i = 0; i < m; i++)
    {
        int index = 0;
        for (int k = 0; k < m; k++)
        {
            if (temp[i] == key[k])
            {
                key[k] = ' ';
                index = k;
                break;
            }
        }
        for (int j = 0; j < row; j++)
        {
            str.push_back(mat[j][index]);
        }
    }
    cout << "Encrypted String:-  " << str << endl;
}
void Decrypt(string &str, string key)
{
    cout << "Encrypted String:-  " << str << endl;
    int n = str.length();
    int m = key.length();
    int row = n / m;
    if (n % m != 0)
    {
        row++;
    }
    char mat[row][m];
    int k = 0;
    string temp = key;
    sort(temp.begin(), temp.end());
    for (int i = 0; i < m; i++)
    {
        int index = 0;
```

```
            for (int k = 0; k < m; k++)
            {
                if (temp[i] == key[k])
                {
                    key[k] = ' ';
                    index = k;
                    break;
                }
            }
            for (int j = 0; j < row; j++)
            {
                mat[j][index] = str[k++];
            }
        }
        str.clear();
        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < m; j++)
            {
                str.push_back(mat[i][j]);
            }
        }
        cout << "Decrypted String:-  " << str << endl;
    }
    int main()
    {

        string str;
        cout << "Enter the string:-  ";
        getline(cin, str);
        string key;
        cout << "Enter the key:-  ";
        cin >> key;
        bool flag = false;
        while (1)
        {
            int option;
            cout << "1. Encrypt\n2. Decrypt\n3. Exit\n";
            cin >> option;
            if (option == 1)
            {
                flag = true;
                Enycrypt(str, key);
            }
            else if (option == 2)
            {
                if (!flag)
                {
```

```
                cout << "Please Encrypt the string first\n";
                continue;
            }
            Decrypt(str, key);
            flag = false;
        }
        else if (option == 3)
        {
            break;
        }
        else
        {
            cout << "Invalid Option\n";
        }
    }
    return 0;
}
```

OUTPUT:

```
PS D:\Programming\CompNetworks Lab 5th Sem> g++ -o cc Transposition_Cipher.cpp
PS D:\Programming\CompNetworks Lab 5th Sem> ./cc
Enter the string:-  Hello everyone welcome to jamia millia islamia
Enter the key:-  Delhi
1. Encrypt
2. Decrypt
3. Exit
1
Given String:-  Hello everyone welcome to jamia millia islamia
Encrypted String:-  H ywm alsaeeoeej il leectmi m or ooilii lvnl amaa
1. Encrypt
2. Decrypt
3. Exit
2
Encrypted String:-  H ywm alsaeeoeej il leectmi m or ooilii lvnl amaa
Decrypted String:-  Hello everyone welcome to jamia millia islamia
1. Encrypt
2. Decrypt
3. Exit
3
PS D:\Programming\CompNetworks Lab 5th Sem>
```

LAB 3:

AIM: Write a Program to implement Playfair Cipher.

```
#include <bits/stdc++.h>
using namespace std;

void generate(vector<vector<char>> &mat, string key)
```

```
{

    int arr[26] = {0}, k = 0, j = 0;
    for (int i = 0; i < key.size(); i++)
    {
        if (key[i] == 'j')
        {
            key[i] = 'i';
        }
        if (arr[key[i] - 'a'] == 0)
        {
            mat[j][k] = key[i];
            k++;
            if (k == 5)
            {
                j++;
                k = 0;
            }
            arr[key[i] - 'a'] = 1;
        }
    }
    for (char i = 'a'; i <= 'z'; i++)
    {
        if (i == 'j')
        {
            i++;
        }
        if (arr[i - 'a'] == 0)
        {
            mat[j][k] = i;
            k++;
            if (k == 5)
            {
                j++;
                k = 0;
            }
            arr[i - 'a'] = 1;
        }
    }
}
void search1(vector<vector<char>> mat, int &i1, int &j1, char temp)
{
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (mat[i][j] == temp)
            {
```

```
                i1 = i;
                j1 = j;
                return;
            }
        }
    }
}
void enycrypt(vector<vector<char>> mat, string &plaintext)
{
    if (plaintext.size() % 2)
        plaintext.push_back('x');

    for (int i = 0; i < plaintext.size(); i = i + 2)
    {
        char a = plaintext[i], b = plaintext[i + 1];
        if (plaintext[i] == plaintext[i + 1])
        {
            b = 'x';
        }
        int i1 = 0, i2 = 0, j1 = 0, j2 = 0;
        search1(mat, i1, j1, a);
        search1(mat, i2, j2, b);
        plaintext[i] = mat[i1][j2];
        plaintext[i + 1] = mat[i2][j1];
    }
}
void decrypt(string &plaintext)
{
    for (int i = 0; i < plaintext.size(); i++)
    {
        if (plaintext[i] == 'x' && 0 < i && i < plaintext.size() - 1)
            plaintext[i] = plaintext[i - 1];
    }
    cout << plaintext[plaintext.size() - 1] << endl;
    if (plaintext[plaintext.size() - 1] == 'x')
        plaintext.pop_back();
}
int main()
{
    vector<vector<char>> mat(5, vector<char>(5, '0'));
    string str, key;
    bool flag = false;
    cout << "Enter the Key text" << endl;
    getline(cin, key);
    cout << "Enter the Plaintext" << endl;
    getline(cin, str);
    generate(mat, key);
    for (int i = 0; i < 5; i++)
```

```
{
    for (int j = 0; j < 5; j++)
    {
        cout << mat[i][j] << " ";
    }
    cout << endl;
}
while (1)
{
    int opt = 0;
    cout << "Select an Option" << endl
         << "Enter 1 for Enycript" << endl
         << "Enter 2 for Decrypt" << endl
         << "Enter 3 for Exit : ";
    cin >> opt;
    if (opt == 1)
    {
        if (flag)
        {
            cout << "Its already Enycrypted" << endl;
            cout << str << endl;
        }
        else
        {
            enycrypt(mat, str);
            cout << "Enycrpted string is" << endl;
            cout << str << endl;
            flag = true;
        }
    }
    else if (opt == 2)
    {
        if (flag == false)
        {
            cout << "Its already Decrypted" << endl;
            cout << str << endl;
        }
        else
        {
            enycrypt(mat, str);
            decrypt(str);
            cout << "Deycrptd string is" << endl;
            cout << str << endl;
        }
    }
    else if (opt == 3)
    {
        break;
```

```
        }
        else
        {
            cout << "Invalid Option\n";
        }
    }
}
```

OUTPUT:

```
PS D:\Programming\CompNetworks Lab 5th Sem> g++ -o cc Playfair_Cipher.cpp
PS D:\Programming\CompNetworks Lab 5th Sem> ./cc
Enter the Key text
networks
Enter the Plaintext
computernetworksprogramminglab
n e t w o
r k s a b
c d f g h
i l m p q
u v x y z
Select an Option
Enter 1 for Enycript
Enter 2 for Decrypt
Enter 3 for Exit : 1
Enycrpted string is
hnpmxnnkenwtnbskiawharmxindpba
Select an Option
Enter 1 for Enycript
Enter 2 for Decrypt
Enter 3 for Exit : 2
b
Deycrptd string is
computernetworksprogramminglab
Select an Option
Enter 1 for Enycript
Enter 2 for Decrypt
Enter 3 for Exit : 3
PS D:\Programming\CompNetworks Lab 5th Sem>
```

LAB 4:

AIM: Write a Program to implement Vignere Cipher.

```
#include <bits/stdc++.h>
using namespace std;

void Table(char table[26][26])
{
    for (int i = 0; i < 26; i++)
    {
        for (int j = 0; j < 26; j++)
```

```
        {
            table[i][j] = (i + j) % 26 + 'A';
        }
    }
}
void Enycrypt(string &text, string key, char table[26][26])
{
    cout << "Given String:-  " << text << endl;
    for (int k = 0; k < text.size(); k++)
    {
        cout << k << " ";
        int i = text[k] - 'A', j = key[k] - 'A';
        if (text[k] >= 'A' && 'Z' >= text[k])
        {
            text[k] = table[i][j];
        }
    }
    cout << "Encrypted String:-  " << text << endl;
}
void Decrypt(string &text, string key, char table[26][26])
{
    cout << "Encrypted String:-  " << text << endl;

    for (int k = 0; k < text.size(); k++)
    {
        int j = key[k] - 'A';
        if (text[k] >= 'A' && 'Z' >= text[k])
        {
            for (int i = 0; i < 26; i++)
            {
                if (table[i][j] == text[k])
                {
                    text[k] = i + 'A';
                    break;
                }
            }
        }
    }
    cout << "Decrypted String:-  " << text << endl;
}
int main()
{
    string text;
    cout << "Enter the string:-  ";
    getline(cin, text);
    string key;
    cout << "Enter the key:-  ";
    cin >> key;
```

```
    string temp = key;
    while (key.size() < text.size())
        key += temp;
    bool flag = false;
    char table[26][26];
    Table(table);
    for(int i=0;i<text.size();i++){
        if(text[i]>='a'&&text[i]<='z'){
            text[i]=text[i]-'a'+'A';
        }
    }
    for(int i=0;i<key.size();i++){
        if(key[i]>='a'&&key[i]<='z'){
            key[i]=key[i]-'a'+'A';
        }
    }
    while (1)
    {
        int option;
        cout << "1. Encrypt\n2. Decrypt\n3. Exit: \n";
        cin >> option;
        if (option == 1)
        {
            flag = true;
            Enycrypt(text, key, table);
        }
        else if (option == 2)
        {
            if (!flag)
            {
                cout << "Please Encrypt the string first\n";
                continue;
            }
            Decrypt(text, key, table);
            flag = false;
        }
        else if (option == 3)
        {
            break;
        }
        else
        {
            cout << "Invalid Option\n";
        }
    }
}
```

OUTPUT:

```
PS D:\Programming\CompNetworks Lab 5th Sem> g++ -o cc Vignere_Cipher.cpp
PS D:\Programming\CompNetworks Lab 5th Sem> ./cc
Enter the string:-  Computer Networks
Enter the key:-  Program
1. Encrypt
2. Decrypt
3. Exit:
1
Given String:-  COMPUTER NETWORKS
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 Encrypted String:-  RFAVLTQG BKKWAGBG
1. Encrypt
2. Decrypt
3. Exit:
2
Encrypted String:-  RFAVLTQG BKKWAGBG
Decrypted String:-  COMPUTER NETWORKS
1. Encrypt
2. Decrypt
3. Exit:
3
PS D:\Programming\CompNetworks Lab 5th Sem>
```

LAB 5:

AIM: Write a Program to implement TCP Server & Client.

→Server:-

```python
import socket

# Define the host and port to listen on
HOST = '127.0.0.1'
PORT = 12345

# Create a socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to a specific address and port
server_socket.bind((HOST, PORT))

# Listen for incoming connections
server_socket.listen()

print(f"Server is listening on {HOST}:{PORT}")

# Accept a connection from a client
client_socket, client_address = server_socket.accept()
print(f"Accepted connection from {client_address}")
```

```
# Receive and print messages from the client
while True:
    data = client_socket.recv(1024).decode('utf-8')
    if not data:
        break
    print(f"Received from client: {data}")

    # Send a response back to the client
    response = f"Server received: {data}"
    msg=input("Enter message to send (type 'exit' to quit): ")
    client_socket.send(msg.encode('utf-8'))

# Close the connection
client_socket.close()
server_socket.close()
```

→Client:-

```
import socket

# Define the server address and port
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 12345

# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server
client_socket.connect((SERVER_HOST, SERVER_PORT))
print(f"Connected to server at {SERVER_HOST}:{SERVER_PORT}")

# Send messages to the server
while True:
    message = input("Enter message to send (type 'exit' to quit): ")
    if message.lower() == 'exit':
        break
    client_socket.send(message.encode('utf-8'))

    # Receive and print the response from the server
    response = client_socket.recv(1024).decode('utf-8')
    print(f"Server response: {response}")

# Close the connection
client_socket.close()
```

OUTPUT:

```
PS C:\Users\hp\Desktop\web dev> python -u "c:\Users\hp
\Desktop\web dev\TCPServer.py"
Server is listening on 127.0.0.1:12345
Accepted connection from ('127.0.0.1', 60881)
Received from client: Hello
Enter message to send (type 'exit' to quit): hello
Received from client: how are you
Enter message to send (type 'exit' to quit): i am fine
 what about you
Received from client: i am also fine
Enter message to send (type 'exit' to quit):
```

```
PS C:\Users\hp\Desktop\web dev> python .\TCPClient.py

Connected to server at 127.0.0.1:12345
Enter message to send (type 'exit' to quit): Hello
Server response: hello
Enter message to send (type 'exit' to quit): how are
you
Server response: i am fine what about you
Enter message to send (type 'exit' to quit): i am als
o fine
```

LAB 6:

AIM: Write a Program to implement UDP Server & Client.

→Server:-

```python
import socket

# Define the host and port to bind to
HOST = '127.0.0.1'
PORT = 12345

# Create a UDP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind the socket to a specific address and port
server_socket.bind((HOST, PORT))

print(f"UDP Server is listening on {HOST}:{PORT}")

while True:
    # Receive data and address from the client
    data, client_address = server_socket.recvfrom(1024)
    print(f"Received from {client_address}: {data.decode('utf-8')}")

    # Send a response back to the client
    response = f"Server received: {data.decode('utf-8')}"
    msg=input("Enter message to send (type 'exit' to quit): ")
    server_socket.sendto(msg.encode('utf-8'), client_address)
```

→Client:-

```python
import socket
```

```python
# Define the server address and port
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 12345

# Create a UDP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    # Get user input and send it to the server
    message = input("Enter message to send (type 'exit' to quit): ")
    if message.lower() == 'exit':
        break

    client_socket.sendto(message.encode('utf-8'), (SERVER_HOST, SERVER_PORT))

    # Receive and print the response from the server
    response, _ = client_socket.recvfrom(1024)
    print(f"Server response: {response.decode('utf-8')}")

# Close the socket
client_socket.close()
```

OUTPUT:

```
PS C:\Users\hp\Desktop\web dev> python -u "c:\Users\hp
\Desktop\web dev\UDPServer.py"
UDP Server is listening on 127.0.0.1:12345
Received from ('127.0.0.1', 49343): hello
Enter message to send (type 'exit' to quit): how are y
ou bhai
Received from ('127.0.0.1', 49343): i am fine bro
Enter message to send (type 'exit' to quit): where are
 you
Received from ('127.0.0.1', 49343): i am in the lab
Enter message to send (type 'exit' to quit): ok i will
 text you back
```

```
PS C:\Users\hp\Desktop\web dev> python .\UDPClient.p
y
Enter message to send (type 'exit' to quit): hello
Server response: how are you bhai
Enter message to send (type 'exit' to quit): i am fi
ne bro
Server response: where are you
Enter message to send (type 'exit' to quit): i am in
 the lab
Server response: ok i will text you back
Enter message to send (type 'exit' to quit): exit
PS C:\Users\hp\Desktop\web dev>
```

GROUP PROJECT

→Server:-

```
import random
import socket
import threading
import json

print("\033c")

PORT = 4000
# size of data in bytes that can go in one packets
HEADER = 1024
FORMAT = "utf-8"
MAX_CLIENT = 2
DISCONNECT_MESSAGE = "!DISCONNECTED!"
FIRST_CONNECTION = "!FIRST_CONNECTION!"
SERVER = '192.168.56.1'
ADDRESS = (SERVER, PORT)
MAX_SIZE = 1000001

# stores the client information like username
user_list = {}
list_of_keys = {}

#########################################################################
def decrypt_message(s, key):
    ans = ""
    mat = [[' ' for _ in range(key)] for _ in range((len(s) - 1) // key + 1)]
    k = 0
    n=int((len(s) - 1) // key + 1)
    for i in range(key):
        for j in range(n):
            if k < len(s):
                mat[j][i] = s[k]
                k += 1

    for i in range(n):
        for j in range(key):
            ans += mat[i][j]

    return ans
#########################################################################

"""
Here we made a socket instance and passed it two parameters. The first
parameter is AF_INET and the second one is SOCK_STREAM. AF_INET refers to the
address-family ipv4. The SOCK_STREAM means connection-oriented TCP protocol.
```

```
"""
try:
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error as err:
    print(f"[UNABLE TO CREATE SOCKET] : {err}...\n")
    exit(0)

"""
A server has a bind() method which binds it to a specific IP and port so that
it can listen to incoming requests on that IP and port.
"""
try:
    server.bind(ADDRESS)
except socket.error as err:
    print(f"[UNABLE TO BIND TO THE SPECIFIC IP AND PORT] : {err}...\n")
    exit(0)

# send message to the client

def sendMessage(msg, client_connection, client_address):
    try:
        global user_list
        global list_of_keys
        client_object = json.loads(msg)
        name=client_object['Reciever']
        if name.lower() != "all":
          clco = list_of_keys.get(name, {}).get('client_connection', None)
          if clco:
             clco.send(msg.encode(FORMAT))
          else:
           temp={"name":"Server","decrypted":f"Unable to send message to
{name}"}
           errmessage=json.dumps(temp)
           client_connection.send(errmessage.encode(FORMAT))
        else:
            for key,data in list_of_keys.items():
               cl_conn=data.get('client_connection',None)
               if cl_conn:
                cl_conn.send(msg.encode(FORMAT))

    except socket.error as err:
        #global user_list
        print(
            f"[UNABLE TO SEND MESSAGE TO THE
{user_list[client_address]['name']}] : {err}...\n")
        del user_list[client_address]
        # exit the helper thread created not the main thread
        exit(0)
```

```
# decode the message if it was the first message or the other message and
respond accordingly

def decodeMessage(str, client_connection, client_address):
    client_object = json.loads(str)
    if client_object['msg'] == FIRST_CONNECTION:
        global user_list
        global list_of_keys
        user_list[client_address] = {
            "name":  client_object['name'],
        }
        list_of_keys[client_object['name']] = {
            "client_address": client_address,
            "client_connection": client_connection,
        }
        return f"joined the server."
    else:
        msg = client_object["msg"]
        dcrptd_msg = client_object["decrypted"]
        print(msg)
        sendMessage(str, client_connection, client_address)
        return dcrptd_msg

# handle's client queries

def handleClient(client_connection, client_address):
    print(f"[NEW CONNECTION] {client_address} connected.\n")
    global user_list
    connected = True
    while connected:
        # reciveing response from client
        try:
            str = client_connection.recv(HEADER).decode(FORMAT)
        except socket.error as err:
            print(
                f"[UNABLE TO RECIVE MESSAGE FROM THE
{user_list[client_address]['name']}] : {err}...\n")
            del user_list[client_address]
            # exit the helper thread created not the main thread
            exit(0)

        if len(str) == 0:
            continue

        msg = decodeMessage(str, client_connection, client_address)
        if msg == DISCONNECT_MESSAGE:
```

```
            # disconnect the client from the server if message is
!DISCONNECTED!
            connected = False
            print(f"{user_list[client_address]['name']} is offline now.")
            continue
        if client_address in user_list and 'key' in user_list[client_address]:
         msg = decrypt_message(msg, int(user_list[client_address]['key']))
        print(f"{user_list[client_address]['name']} : {msg}")

    # removing the client from the list after he/she get disconnected
    del user_list[client_address]
    client_connection.close()

def start():
    """
    A server has a listen() method which puts the server into listening mode.
This allows the server to listen to incoming connections.
    """
    server.listen(MAX_CLIENT)
    print(f"[LISTENING]  server is listening on {SERVER}\n")
    connected = True
    while connected:
        """
        And last a server has an accept() and close() method. The accept
method initiates a connection with the client and the close method closes the
connection with the client.
        """
        try:
            client_connection, client_address = server.accept()
        except socket.error as err:
            print(f"[UNABLE TO CONNECT TO THE CLIENTS] : {err}...\n")
            exit(0)

        try:
            thread = threading.Thread(target=handleClient, args=(
                client_connection, client_address))
            thread.start()
        except socket.error as err:
            print(f"[UNABLE TO CREATE THREAD] : {err}...\n")
            exit(0)

        # -1 bcoz one thread is running the server
        print(f"[ACTIVE CONNECTIONS] {threading.active_count()-1}\n")

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

→Client:-

```
import socket
import json
import threading
import time

print("\033c")

PORT = 4000
HEADER = 1024
FORMAT = "utf-8"
DISCONNECT_MESSAGE = "!DISCONNECTED!"

SERVER = "192.168.56.1"
ADDRESS = (SERVER, PORT)

def encrypt_message(s, key):
    ans = ""
    n=int((len(s) - 1) // key + 1)
    mat = [[' ' for _ in range(key)] for _ in range((len(s) - 1) // key + 1)]
    k = 0

    for i in range(n):
        for j in range(key):
            if k < len(s):
                mat[i][j] = s[k]
                k += 1

    for i in range(key):
        for j in range(n):
            ans += mat[j][i]

    return ans


def decrypt_message(s, key):
    ans = ""
    mat = [[' ' for _ in range(key)] for _ in range((len(s) - 1) // key + 1)]
    k = 0
    n=int((len(s) - 1) // key + 1)
    for i in range(key):
        for j in range(n):
            if k < len(s):
                mat[j][i] = s[k]
                k += 1

    for i in range(n):
        for j in range(key):
```

```
            ans += mat[i][j]

    return ans


try:
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error as err:
    print(f"[UNABLE TO CREATE SOCKET] : {err}...\n")
    exit(0)

try:
    client.connect(ADDRESS)
except socket.error as err:
    print(f"[UNABLE TO CONNECT TO THE SERVER] : {err}...\n")
    exit(0)


def sendMessage(msg, key,reciever,user_name):
    encrypted_msg = encrypt_message(msg, key)
    json_object = {'msg': encrypted_msg, "key": key, "decrypted":
msg,"Reciever":reciever,"name":user_name}
    msg = json.dumps(json_object)
    try:
        client.send(msg.encode(FORMAT))
    except socket.error as err:
        print(f"[UNABLE TO SEND MESSAGE TO THE SERVER] : {err}...\n")
        exit(0)

def somemessagedeleter(msg):
    offensive_words = ["dummy1", "dummy2", "dummy3", "dummy4", "dummy5",
"dummy6", "dummy7", "dummy8", "dummy9"]
    replace_words ={
        "jamia":" jamia millia islamia ",
        "delhi":" New Delhi ",
        "Engg":" Engineering ",
    }
    for word in offensive_words:
        if word in msg:
            msg = msg.replace(f" {word} ", " ")
            msg=msg.replace(f" {word}", " ")
            msg=msg.replace(f"{word} ", " ")
    for key,value in replace_words.items():
        if key in msg:
            msg = msg.replace(f" {key} ",value)
            msg=msg.replace(f" {key}",value)
            msg=msg.replace(f"{key} ",value)
    return msg
```

```
def receiveMessage():
    try:
        # timeout = 60
        # client.settimeout(timeout)
        server_msg = client.recv(HEADER).decode('utf8')
        obj = json.loads(server_msg)
        msg=obj['decrypted']
        sender=obj['name']
        msg=somemessagedeleter(msg)
        print(f"\n{sender}",": ",f"{msg}\n\n""Enter The Text: ")
        # message_sender()
        return server_msg
    # except socket.timeout:
    #     print("Timeout: No message received within the specified time.")
    #     return None
    except socket.error as err:
        print(f"[UNABLE TO RECEIVE MESSAGE FROM THE SERVER] : {err}...\n")
        exit(0)
    # finally:
    #     client.settimeout(None)

def message_sender():
    while True:
        text = input("Enter The Text: ")
        recv=input("Enter The Reciever Name: ")
        sendMessage(text, key,recv,user_name)
        time.sleep(2)

def message_receiver():
    while True:
        server_msg = receiveMessage()
        time.sleep(2)

user_name = input("Enter your name : ")
json_object = {'name': user_name, 'msg': '!FIRST_CONNECTION!'}
msg = json.dumps(json_object)
client.send(msg.encode(FORMAT))

connected = True
key = int(input("Enter The Key: "))

# Create threads for sender and receiver functions
sender_thread = threading.Thread(target=message_sender)
receiver_thread = threading.Thread(target=message_receiver)

# Start both threads
sender_thread.start()
receiver_thread.start()
```

```
# Wait for both threads to finish (this won't happen in this example)
sender_thread.join()
receiver_thread.join()

# Closing the connection from the server
print("Connection Closed!")
client.close()
```
