1. **Client Info buffer** :

Structure :

```
struct clientInfo{
        char name[10], time[50];
        int id, online, fd;  //Random generated 'id' . Online flag 'online' 0 if  not online. 'fd' socket file descriptor.
};
```

2. **Shared memory for the for the Client Info** :

Structure :

```
struct sharedmem_Info{
        struct clientInfo *CI;
        sem_t *lock;
        int *len;
        int idci, idl, idlen;
};
```

Shared between : server and processes that handles the client request.

Lock : All searching and updation requires a prior locking.

3. **Message Info buffer** :

Structure :

```
struct messageBuffer{
        int f; //Whether the receiver is an id or a name
        char from[10], to[10]; // Sender and receiver of the mesage
        char message[MAXLEN]; //Message
};
```

4. **Shared Memory for the Message info buffer** :

Structure :

```
struct sharedmem_Message{
        struct messageBuffer *Q;
        int *head, *tail;
        sem_t *lock;
        int idq, idh, idt, idl;
};
```

Shared Between : Server and the sub processes that handles client requests.

Lock : All updation requires a prior locking.

Client Process :

1. Starts with command ./client (address) (port).
2. After starting creates a socket and tries to connect with the server. If everything is fine, connection is established and client gets a “unique” name and id. which it stores.
3. Client creates a thread so that it can read from standard input and the socket at the same time.
4. When process ends(SIGQUIT) it sends a ‘-q’ command to the server which is an indication that client is going to quit.

Commands in the client process :

l command :

usage :

1. l
   function : Gets the list of all the clients that are online.
   output : Prints name’s and id’s of the clients that are online.
2. l -(N)
   function : checks whether the client with name ’N’ is online.
   output : if it founds the client with name N online the returns his name and id else prints “No client is online.”

m command :

usage :

m -(N) M

N is name or id of the receiver(purely numeric N means N is id otherwise it is name of the receiver)
M is the message.

function : sends the message M to the receiver N.

Output’s :

1. sender should not be same as the receiver.
   output : “Sender same as receiver”.
2. message should be non null.
   output : “No message”.
3. N should be alphanumeric.
   output : “Invalid user”.
4. format of the command is wrong.
   output : “Invalid format”.
5. If the message cannot be sent to the receiver.
   output : “Can’t send message to : [N]”

Server Process :

1. Starts with the command ./server (port).
2. Allocates the shared memory buffer (2., 4. in structure).
3. creates a socket for listening to the client requests.
4. when Client request comes then it checks whether number of clients that are online is less than 5. if yes then it dos the following tasks :
   a. creates a random unique id, name and gets the current time.
   b. stores the info. from a. and client socket file descriptor in the client info buffer(1.)
   c. also sends info. from a. to the client.
5. Forks a new child process to handle the requests of the client.
6. starts listening to the new requests.

Server process that handles a client :

1. if '-l' request comes from the client then it checks which client are online and the sends their information to the client.
2. if '-m' request comes from the client then it check whether message queue is full or not. if yes then replies "Message queue full" to the client else enqueues the message into the message queue.
3. if '-q' requests comes from the client then it makes the client offline and sends message to each client "server : [client] Has gone offline".

Server thread to handle message queues:

1. Dequeues a message from the Message info buffer(3.).
2. Checks whether the receiver is online or not by checking the client info Buffer(1.).
3. if found online the then uses the client's socket file descriptor to send message to the receiver. else it sends "Can't send message to : [R]" to the sender.
4. if the sender of the message is not the server then it logs the message in the log file as follows: 'sender' 'receiver' MESSAGE_SIZE'

Python Scripts :

1. **16CS60R64_loadanalysis.py**
   function : reads the log file and generates characterLoad.csv(number of characters sent by a client till now) and messageLoad.csv(number of message between a pair of client's).

2. **16CS60R64_plot.py**
   function : reads characterLoad.csv and messageLoad.csv files and generates a bar chart for the first file and a heat map for the second file.