Documentation

1. Possible ideas to implement P2P:

1. Central discovery server approach – Most common and easy way is to use a central discovery server, which will provide a list of peers that are currently online. In order for this application to work, the peers must contact the discovery server regularly and update the connectivity information. If no communication is received from a peer within certain amount of time the peer is considered no longer active and the peer record is removed from the server.
2. When a peer wants to contact another peer first, it contacts the discovery server, which will provide a list of peers that are online.
3. Reading a list of super peers from some location and trying to contact them directly.
4. The connectivity information that you need depends on the way you are connecting with the peer. It typically includes information such as port number and IP address.
5. When a peer leaves the network, an updated list is published to all the peers present in the network. After this list is received by every peer, the peer would discontinue attempting to communicate with those peers present in the system.
6. Other than this modification in your previous chat application where you need to change from client-server architecture to P2P, other specifications such as login, logout, creating chat rooms and GUI must have the same protocols as specified in the first project documentation.

Idea chosen:

* We have a central server for client. Client would connect to get information about all clients present in the P2P system.
* In order to be a part of the system, the client must login using the ID and password.
* When a client logs out of the system, it will send a message to the server. This server will disseminate this information to all the peers present in the system.
* After client successfully logs in into the system, the client makes it available to other peers present in the system. It does so by creating a listP2P and sending a message to all the clients (peers) present in the system.
* Here, the process takes place without the use of a server.
* For any client to chat with any other client, it will be provided with an IP address and a port number associated with the client who we want to chat with.
* class Client :
  + The client logs in into the system and gets the list of all clients from the server.
  + We have 2 classes to take care of list of clients: P2PClient, P2PList
  + P2PClient.java has the IP address and port number of the client we want to chat with.
  + P2PList.java is used to save list of all clients present in the system.
  + After client successfully logs into the system, the client gets the list of call clients from the server and the new client will be saved in P2PList.
  + If the client wants to send a message to another client, it will fetch the IP and Port number of the desired client from the P2PClient. After obtaining the IP and port number, it will send the message.
  + Other classes – App.java, Chat.java, ChatRoom.java, Logon.java and other remaining classes – is the same as in previous project.
* class Server
  + Here, the Server is almost same as the Server class in the previous project.
  + The difference is that the Server class in this project is used to just store the messages sent by the client. The messages are stored in order to verify the transmission of the message.
  + It also sends the status of the client. If any client goes offline, this event is broadcasted to everyone in the system.

2.

3. The discovery of the clients in this project is done by implementing a light server. This server is used to notify clients of arrival or departure of any client(s) in the system. The server is implemented to validate the broadcast of messages from one client to another.

4.