Software Requirements Specification (SRS) Document

<P2P Voice call, Lasya Priyanka.M, Sundhar, Venkatesh.B>

Brief problem statement :

Our application allows the users to communicate with each other via chat or voice call when they are in the same network without any charges. Using our Application, you can discover and connect to other devices when each device is in the same network and connected to server, then you can communicate over a speedy connection across distances much longer than a Bluetooth connection. In extension we will try to do file sharing among users, just like photo, video sharing application.

System requirements:

- ★ Our application will run on any android device whose android version is greater than 4.0.
- ★ Our assumption is that the users have a server which is hosted by us.
- ★ Server will have a minimum storage of 1TB.
- ★ Our web application will be implemented in python, javascript, java, xml ,android studio. We use python for backend.

Users profile:

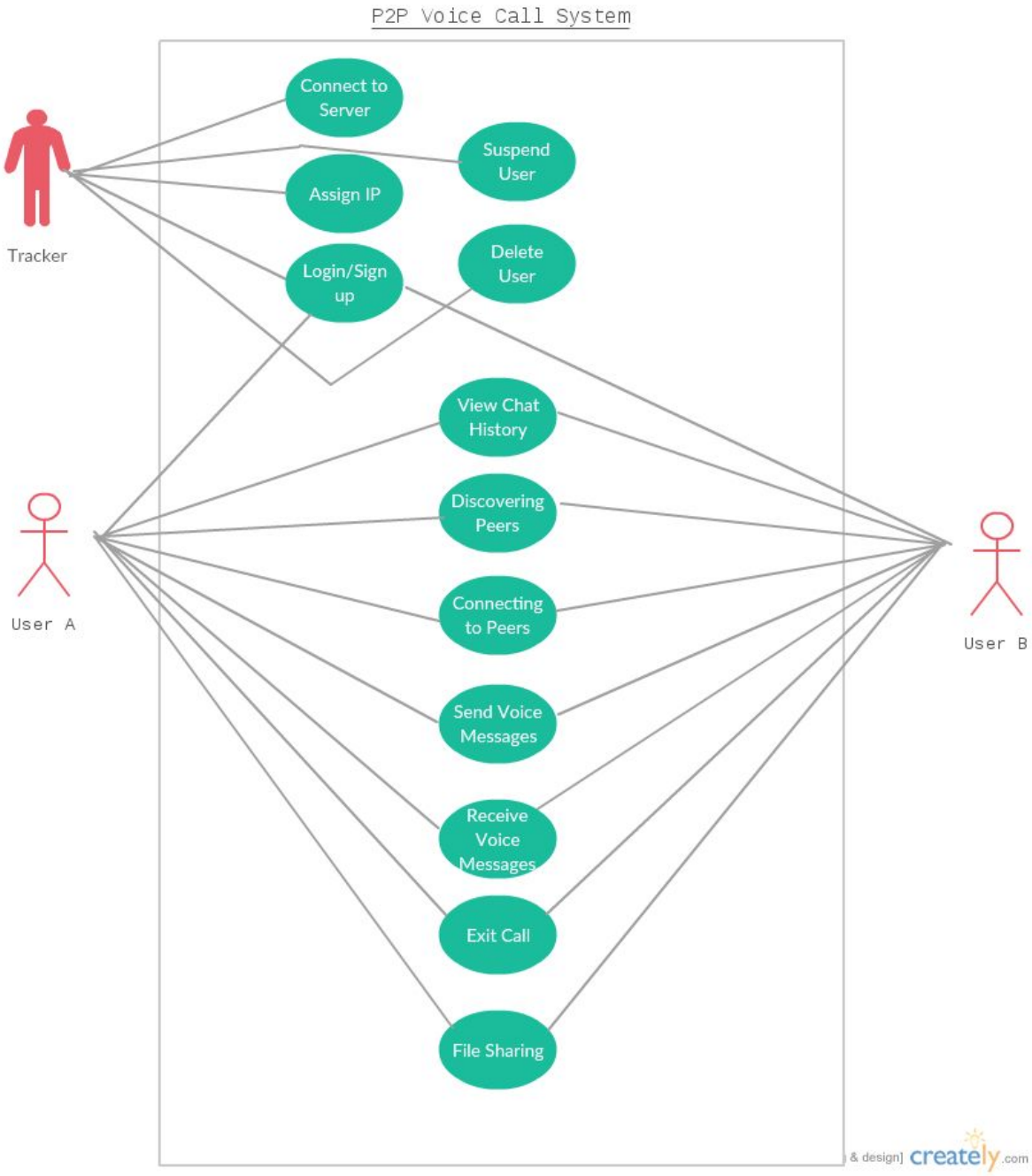
- ★ Users must be there in a same connected network.
- ★ Users should know how to read english and understand it.
- ★ User should know how to use and operate a smartphone.
- ★ User should be able to respond to popups generated on the screen correctly.

Feature requirements (described using use cases)

No.	User case name	Description	Release
1.	Connect to server	Users should first connect to the server and then only they can do further activities.	R1
2.	Login/Sign up	After the application opens, it asks the user to register and login. ➔ If the user is already a member, he logins. ➔ If it is a new user, he registers by clicking sign up.	R1
3.	Assign IP	When the user logins, the tracker now should assign an IP to each user. Hence the socket connection between the client and server is established.	R1
4.	Delete user	If the user is not their in the same connected network anymore, the tracker should delete the user account.	R4
5.	Suspend user	This feature helps the tracker to temporarily halt the user to continue his connection with other user.	R4
6.	View chat history	User will be able to look at the chat history with other clients.	R4
7.	Discovering Peers	User can detect the available connected peers in the range and	R2
8.	Connecting to Peers	User chooses the peer that he wants to make a voice call and establishes a client-client connection.	R2
9.	Send Voice messages	User will be able to send messages to the person with whom he had a socket connection.	R3
10.	Receive Voice	Both the clients will be able to receive	R3

	messages	messages send to them.	
11.	Exit Call	After the conversion is done,both the users should be able to exit the call.	R4
12.	Share files	With this, the users will be able to share files among them.	R5(Extension)

Use case diagram:


P2P Voice Call System
Connect to Server
Suspend User
Assign IP
Tracker
Login/Sign up
Delete User
View Chat History
Discovering Peers
User A
User B
Connecting to Peers
Send Voice Messages
Receive Voice Messages
Exit Call
File Sharing
& design] creately.com

Use case description:

Use Case Number:	UC-01
Use Case Name:	Connect to server
Overview:	There will a server socket. This socket waits for a connection from a client or tracker. Whoever wants to make a call he should first connect to the server. Even tacker also.
Actors:	Tracker, User A and User B <<Clients>>
Pre condition:	Before using this system, user should be from the connected network(same company etc). User must ensure that he have a smartphone and that his device supports this system.
Flow:	Main (success) Flow: 1. User opens his smart phone and open this application. 2. He connects to the server.
	Alternate Flows: If the user couldn't connect to the server, then he is not in the connected network.
Post Condition:	User is successfully connects to the server. Now he can do common actions such as discovering and connecting to peers.

Use Case Number:	UC-02
Use Case Name:	Login
Overview:	When the user opens this application, it would ask the user to register . If he is already a member he just login or type his username.
Actors:	Tracker, User A and User B <<Clients>>
Pre condition:	Users must first connect to the server to login.
Flow:	Main (success) Flow: New user: 1. User register to the server through his email. 2. User gives a username and password.

	3. User logins. Registered user: Simply logins.
	Alternate Flows: User registers first and then he logins.
Post Condition:	User is successfully logged in and how he can do further actions like view chat history, making voice calls etc.

Use Case Number:	UC-03
Use Case Name:	Assign IP
Overview:	As our app works over sockets, whenever a user connects to the server, the tracker must assign IP address to each user called session IP so that socket connection is established between the server and the client.
Actors:	Tracker
Pre condition:	To assign IP to each user, both the tracker and user must first connect to the server and should login. Now the tracker can assign the session IP to the clients.
Flow:	Main (success) Flow: 1. When the socket connection is established between the server and the client, the tracker now assigns an IP for each client who are connected to the server. 2. Tracker assigned IP.
	Alternate Flows: If the tracker couldn't assign the IP, he checks his connection and if its not connected properly, he reconnects to the server , logins and finishes his job.
Post Condition:	Tracker assigns the IP to the client and now the client can make voice calls.

Use Case Number:	UC-04
Use Case Name:	Delete User
Overview:	If the user is not their in the network anymore, the tracker should delete the user account.
Actors:	Tracker

Pre condition:	To delete user, Tracker must first login and then he can delete the user.
Flow:	Main (success) Flow: 1. Tracker gets the list of all clients. 2. He navigates through the list and clicks on the client that he wants to delete. 3. Tracker clicks on delete option. 4. Confirmation messages pop ups and he click yes.
	Alternate Flows: If the tracker is unable to delete the user, he must check his connection and reconnected again if it's not proper.
Post Condition:	Tracker deletes the client or user successfully. Deleted client is not in the list anymore.

Use Case Number:	UC-05
Use Case Name:	Suspend User
Overview:	If the tracker wants to temporarily prevent the user from continuing his socket connection with another client. He can use this feature and suspend the user.
Actors:	Tracker
Pre condition:	To suspend user, Tracker must first login and then he can disable the client's socket connection.
Flow:	Main (success) Flow: 1. Tracker gets the list of all clients. 2. He navigates through the list and clicks on the client that he wants to suspend. 3. Tracker clicks on suspend user option. 4. Confirmation messages pop ups and he click yes.
	Alternate Flows: If the tracker is unable to suspend the user, he must check his connection and reconnect again if the connection is failed.
Post Condition:	Tracker suspends the client or user from being able to send voice messages. User gets the message as Session expired.

Use Case Number:	UC-06
Use Case Name:	View Chat History
Overview:	This helps the user to look at the chats or voice messages with other clients.
Actors:	User A and User B <<Clients>>
Pre condition:	If the user wants to look at the chat history, he must first login.
Flow:	Main (success) Flow: 1. User navigates to the label chats and click on it. 2. He will get a list of chats with other clients. 3. User click on the chat of particular client that he wants to look at. 4. Now he can read or delete the chats that he want to.
	Alternate Flows: If the user is not able to look at the chat history, then he should check his connection and login again.
Post Condition:	User will be able to look at the chat history with other clients.

Use Case Number:	UC-07
Use Case Name:	Discovering Peers
Overview:	The user should be able to discover peers that are available to connect to, and detect available peers that are in range.
Actors:	User A and User B <<Clients>>
Pre condition:	User should already connect to the server and must be logged in.
Flow:	Main (success) Flow: 1. User presses the button ”Discover peers.” 2. If discovery process succeeds and detects peers, the system obtains the list of peers 3. System server provides the user with the list of all available users that are online.
	Alternate Flows: If the user was unable to receive the available user list , we will display a error message and ask him to do it again.

Post Condition:	User receives the list of available peers. Later the user iterates through to find the peer that he want to connect to.

Use Case Number:	UC-08
Use Case Name:	Connecting to Peers
Overview:	After obtaining a list of possible peers,the user connects to the device that he wants to make a voice call.
Actors:	User A and User B <<Clients>>
Pre condition:	User must be logged in and should also have the list of users that are available online.
Flow:	Main (success) Flow: 1. User makes a function call connect() to connect to the device. 2. This function call contains the information of the device that the user want to connect to. 3. Later the user will be notified with a connection success or failure
	Alternate Flows: If the user is notified by connection failure, he will again call connect() method on the device he wanted to connect to.
Post Condition:	Connection is established between the two users and now they can make voice call.

Use Case Number:	UC-09
Use Case Name:	Send Voice Messages
Overview:	When a connection happens, both the users(clients) can send the data. Meanwhile the tracker keep track of both the devices.
Actors:	User A and User B <<Clients>>
Pre condition:	Both the clients must have a client-server connection and also client-client connection.
Flow:	Main (success) Flow: 1. Once a connection is established, users can send voice messages among each other from the devices through sockets. 2. User records the voice message.

	3. User sends the voice message.
	Alternate Flows: If the user is unable to send messages, he need to go to the previous step and should check if the connection is proper or not.
Post Condition:	Voice messages will be send to respective users and the conversation starts.

Use Case Number:	UC-10
Use Case Name:	Receive Voice Messages
Overview:	After the connection between the two client sockets is established, they both send messages and both the users should be able to receive the voice messages which were sent to them.
Actors:	User A and User B <<Clients>>
Pre condition:	Users should first send voice messages and then only other user can receive.
Flow:	Main (success) Flow: 1. User receives the voice messages from another client. 2. Opens the voice messages.
	Alternate Flows: Even though after sending the messages to client 2, if client2 doesn't receive the messages, he should check his connection.If there is a proper connection, client1 voice messages should be delivered to client2.
Post Condition:	Client device receives the messages from another client and conversation continuous.

Use Case Number:	UC-11
Use Case Name:	Exit Call
Overview:	User makes the necessary conversation with another client and ends the conversation.
Actors:	User A and User B <<Clients>>
Pre condition:	Clients should have made a call priorly.

Flow:	Main (success) Flow: 1. After done with conversation, one of the client ends the call. 2. Press the exit call button. 3. User will be shown with a message “Call Ended”
	Alternate Flows:.If the user s=is not able to end the call, he again press the button End call.
Post Condition:	Users are successfully disconnected from the Voice call.

Extension (if time permits):

Use Case Number:	UC-12
Use Case Name:	File sharing
Overview:	With this feature, users can not only send voice messages they can also carry out actions like sharing videos, pictures, files etc;
Actors:	User A and User B <<Clients>>
Pre condition:	In Order to share files between the users, both the user should first login and must have a socket connection between them.
Flow:	Main (success) Flow: 1. User opens the chat with the other client. 2. Tap the icon at the top of the screen. (attach files icon) 3. Choose the file he wants to send from his device. 4. Selects the desired file to send and tap Send in the popup.
	Alternate Flows: If the unable to share files, they should check their socket connection first and reconnect to the peer that he wants to share files with.
Post Condition:	Files are successfully shared among the clients and the conversation continues till the session expires.