[Video Chat Application]

A Project Report Submitted
in Partial Fulfillment of the Requirements
for **Summer Internship**
Project

Under the Guidance of
Mr. Parag Tiwari
and Supervision of
Pradnya Mohite

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY
Powai, Maharashtra, INDIA
June, 2013

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to Professor Deepak B. Phatak for offering us this internship at IIT Bombay and handling us the responsibility of this project. His willingness to motivate us contributed tremendously to our project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life.

We are thankful to our project's mentor Mr. Parag Tiwari and our guide Ms. Pradnya Mohite, for their exemplary guidance, monitoring and constant encouragement throughout the project. We thank them for their cordial support and valuable information which helped us in completing this task through various stages.

Also, we would like to extend our sincere regards to all the non-teaching staff at IIT Bombay for their timely support. Although, there may be many who remain unacknowledged in this humble note of gratitude, but there are none who remain unappreciated.

Ajay Kumar
Cheena Jain
Manisha Barnwal
Nishant Chaubey
Prashant Aher
Ruchit Gandhi
Sankalp Rangare
Shreshtha Pankaj
Sumantra Chattopadhyay

Contents

CHAPTER NO. 1

SOFTWARE REQUIREMENT SPECIFICATION

1.1 INTRODUCTION

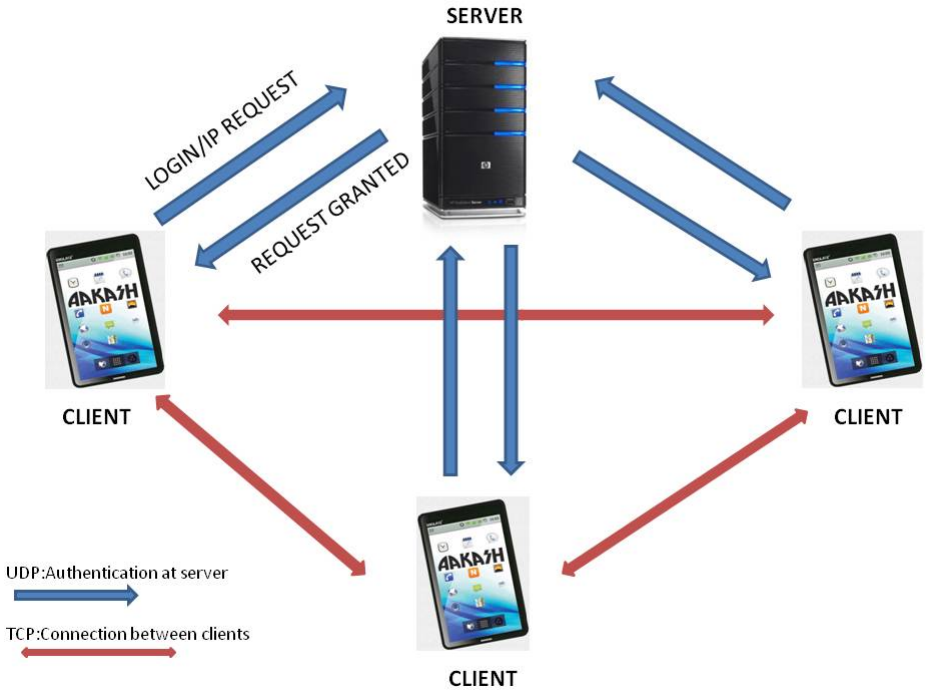
The document aims at defining the overall software requirements for "Video-Chat" and Server side application. Efforts are been made to define the requirement exhaustively and accurately. The final product will be having only features/functionalities mentioned in this document and assumption for any additional functionality/feature should not be made by any of the parties involved in developing /testing/implementing the products. In case it is required to have some additional features, a formal change request will need to be raised and subsequently a new release of this document and/or product will be produced.

1.1.1 Purpose

This specification document describes the capabilities that will be provided by the android applications "Video Chat" and "Server". It also states the various required constraints by which the system will abide. The intended audience for this document is the development team, testing team and the users of the product.

1.1.2 Scope

The software "Video Chat" is an application that will be used by general users to do chat, file share, voice calls and video calls using Aakash tablet. Also audio and chat conferencing facility



has been provided. To make a call both the users need to be registered at the server. This communication is via wi-fi and will incur no expenditure to the end users. Communication between any two users will be enabled as long as both are logged-in at the server.
An application "Server" is required to register the users and maintain the information regarding the MAC and current IP addresses of all the logged-in users. The clients and server must be connected to same network.

1.1.3 Overview

The intended audience for this document is the development team, testing team and the users of the product.

1.2 OVERALL DESCRIPTION

In the present day where communication plays such an important role in day-to-day activity, this product was proposed with the aim to provide free communication over smaller distance eg. within an organization campus. This product is a part of Aakash tablet.
"Video Chat" application can be implemented on any android based device.

1.2.1 Product Functionality

Major functions of the "Video Chat" application:

1. Connect and Login with the desired server.
2. Initiate a communication to any available user in following ways:

   Perform text chat with an online user

   Add more users to the current chat

   Share Files with one or more than one users

   Initiate and receive audio call to and from available user

   Initiate and receive video call to and from available user

Major functions of the "Desktop Server" software:

1. Register the user who wish to use this product and allocate them tablet a unique username.
2. Login and logout the users when they start and close their application respectively.
3. Exchange IP addresses between two available users in the network.

1.2.2 Operating Environment

The application "Video-Chat" will be developed to operate on any android based devices, like mobile phones and tablets with android versions 4.0 (API level 14) or greater. It requires a wi-fi connection facility in the device. For voice communication a speaker and an internal/ external microphone is required.
Server software has to be implemented on desktop computers (windows/ linux based). It requires a WiFi or LAN connection to connect to all the users. It also requires JDK 1.6 or above, mysql server 5.5 and jdbc driver.

1.2.3 User Documentation

A detailed user manual and possibly on-line help will be delivered along with the software.

1.2.4 Assumptions And Dependencies

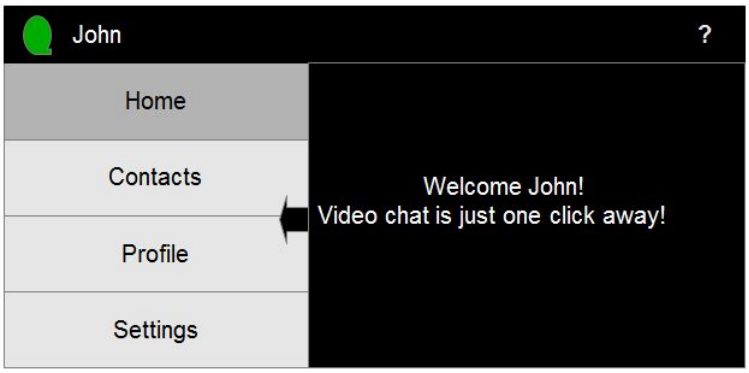
1. JDK 1.6 or later
2. JRE 1.6 or later
3. Eclipse Indigo
4. Android SDK 4.0 or later
5. ADT plugin 18

1.3 SPECIFIC REQUIREMENTS

1.3.1 External Interface Requirements

User Interfaces

User Interfaces For the “Video-Chat” application, the user interface will provide buttons to login, change settings, start call and end call. The application will have its own contact list to save roll no and name of other users.




John
?
Home
Contacts
Profile
Settings
On-line
Off-line


Incoming Call.....
Smith D
Accept
Reject
Chat
Only Audio
Send File

The user will be notified with proper error messages in case of connection errors. The Server software has a simpler user interface. It provides facility to enter a users information for registration. It also shows the list of all registered users and indicate whether they are online/offline. Request from any user is processed automatically by the server software as long as it is running, without any involvement of manual help.

Hardware Interfaces

- Computer and android device/tablet should be connected to a wi-fi access point.
- Speaker and an internal/external microphone are also required with the client device.

Software Interfaces

Software interface for "Video-Chat" application:

- Device based on android operating system versions 4.0(API level 14) or greater.
- Device should provide facility for lightweight Database Management System -SQLite.

Software interface for "Server" application:

- Computer with Windows or Linux based operating system (Database Management System used is MySQL 5.5).

Communication Interfaces

- Product uses UDP protocol for communication between a client and the server.
- TCP protocol is used to establish all kinds of client to client communication.
- UDP protocol is used for transmitting the audio and video data between two users during the call.

1.3.2 Functional Requirements

Major functions of the Video Chat application:

- Authenticate and Login user to the server.
- Initiate Peer-to-Peer Audio call, Video Call, File Share, Group Chat, Audio Conferencing.

- Receive different requests like Audio Call receiving, Video Call receiving, File receiving.
- A Contact List is provided where the user may add additional contacts.
- User may set the frequency for audio communication and IP of the Server.

Major functions of the Server software:

- New users can be registered and unregistered.
- Keeps track of online and offline users .
- Maintains a log of all the client requests.
- View Present state of Database at any point of time.

1.3.3 Performance Requirements

1. Any transaction between a client and the server will take approximately not more than 3 seconds.
2. Establishing a call connection between two clients take approximately not more than 4 seconds.
3. The lag in the audio and video data being transferred between the two users during call will be approximately not more than 3 seconds.
4. Any no. of users may be logged-in at the server at any time.
5. All the users need to be connected to same/different access points of the same network.
6. The “Video-Chat” application should be light to minimise the power consumption of the device.

CHAPTER NO. 2

DESIGN DOCUMENTS

2.1 INTRODUCTION

2.1.1 Background

Aakash, the low cost Indian android-based tablet, can be used in a variety of ways to spread literacy and education in the country. It is an excellent device which can be used by Indian students and teachers to cater their needs. There are many efforts going on in IIT Bombay to make the device more powerful and enhance the utility of the tablet. This project aims at developing an android application focusing on Peer-to-Peer Audio and Video Conferencing, Group Chat, and File Sharing between Aakash tablets through Wi-Fi connectivity since the existing GSM facility involves a cost factor. Our objective was to provide free communication between the tablets by taking help of its Wi-Fi connectivity.

2.1.2 Design Goals

Peer-to-Peer Video Conferencing was our main design goal. It was very important to make the video quality good in this kind of voice communication application. Minimum lag in the transmission was also one of the design goals. Along with this, Audio Conferencing, Group Chat, Group File Sharing were also our goals. This application involves very basic use of server since it is based on Peer-to-Peer Communication.

2.2 PROJECT PLAN

2.2.1 Title and Scope of the Project

Title: Video Conferencing

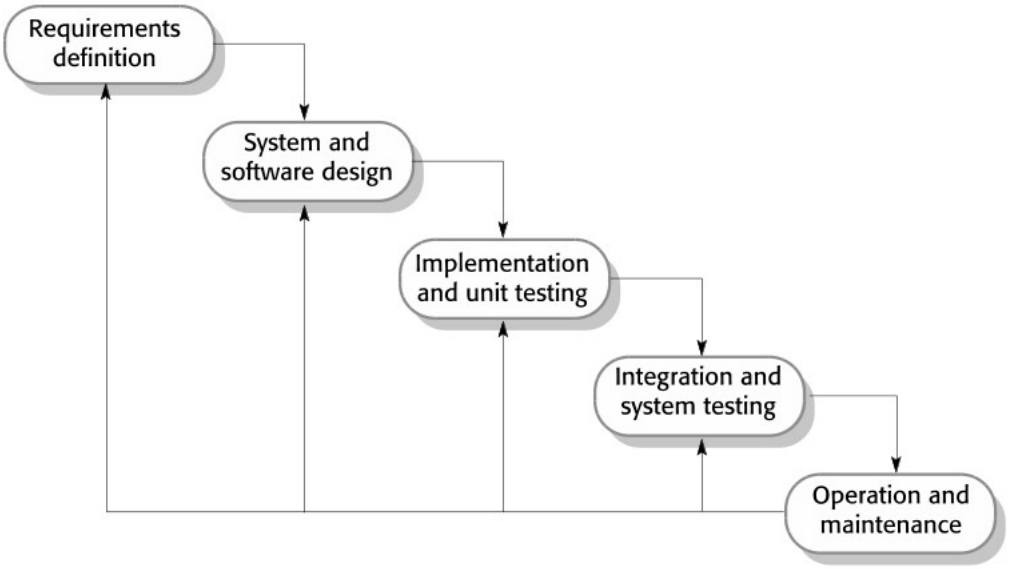
Scope: The software Video Conferencing is an application that will be used by general users for peer-to-peer communication like audio and video conferencing, group chat, file sharing etc using Aakash tablets. To use any functionality of the application, the user needs to be registered at the server. This communication is via WiFi and will incur no expenditure to the end users. Communication between any two users will be enabled as long as both are logged-in at the server. An application Server is required to register the users and maintain the information regarding the MAC, current IP addresses, username, password and availability status of all the logged-in users. The clients may be connected to same/different WiFi given both the routers are registered at the same network. Server must be connected to same WiFi network.

2.2.2 Resource Requirements

- *Hardware Requirements*
  - The Server (Desktop/tablet) and Client (tablet) should have WiFi connectivity.
  - To use the Video Conferencing application a headphone or headset and an internal microphone is required with the device.
- *Software Requirements*
  - Software interface for Video Conferencing application
    - Any device based on android operating system versions 4.0.3 and higher.
    - Support for lightweight Database Management System SQLite.
  - Software interface for Server software :
    - Any computer with windows or linux based operating system (Database Management System used is MySQL).
    - Any device based on android operating system versions 4.0.3 and higher (Database Management system used is SQLite).

2.2.3 Model Used (Iterative Model)

The Iterative lifecycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model , until the product is accepted as shown below:



A Requirements phase - in which the requirements for the software are gathered and analyzed. Iteration should eventually result in a requirements phase that produces a complete and final specification of requirements. A Design phase - in which a software solution to meet the requirements is designed. This may be a new design, or an extension of an earlier design. An Implementation and Test phase - when the software is coded, integrated and tested. A Review phase - in which the software is evaluated, the current requirements are reviewed, and changes and additions to requirements proposed.

2.2.4 Task List

1 . Establish Connection between Server and Client using WiFi.

2 . Establishing Peer to Peer Connection.

3 . Database Management and Database Connectivity.

4 . Parsing the Messages in the server and updating the Database.

5 . Fetching IPs' of all available users from server.

6 . Recording and Playing Audio in Aakash Tablet.

7 . Recording and Playing Video in Aakash tablet.

8 . Transmitting live audio peer-to-peer(Conference).

9 . Transmitting live video peer-to-peer(Call).

10 . Setting up of Calling Functionality between Clients for Video Call.

11 . Setting up of Calling Functionality between Clients for Audio Conference.

12 . Video Call Testing

13 . Audio Conference Call Testing

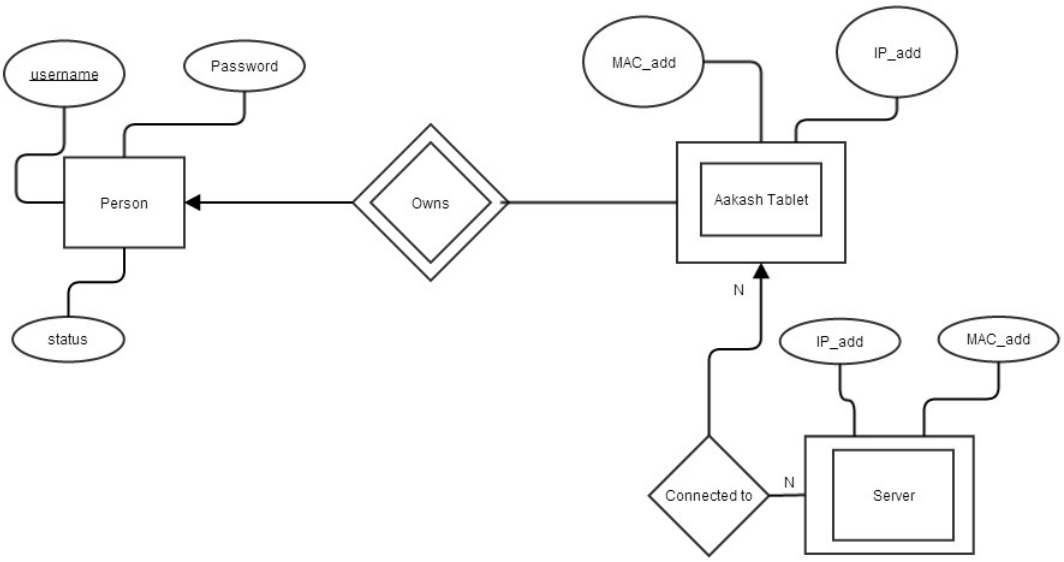
14 . Group Chat Testing along with Group File Sharing

2.3 DESIGN AND IMPLEMENTATION

2.3.1 High Level Design Document

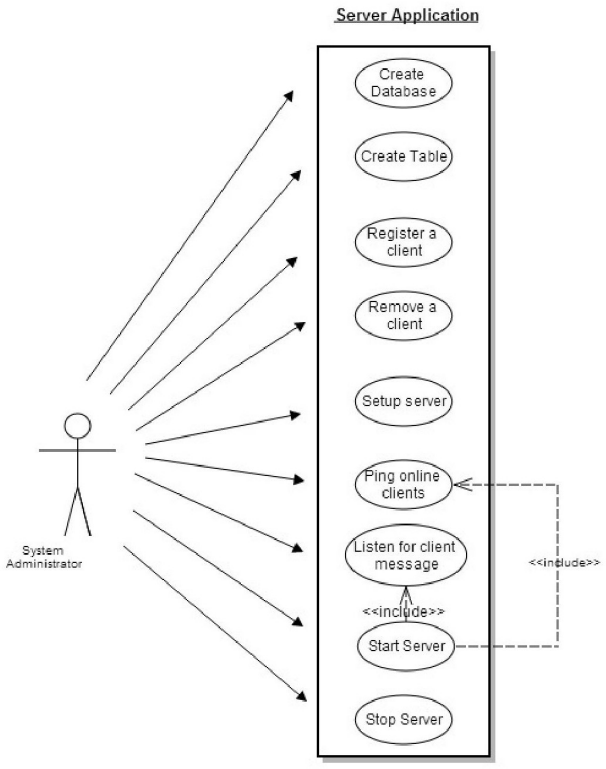
- *E-R Diagram*

Server and Client Side

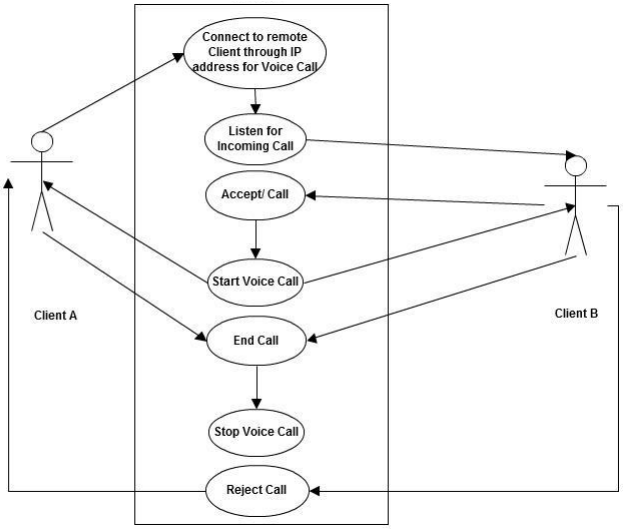


Use Case Diagram

- Server Application
  - Purpose: To maintain a database which has data regarding the clients and to register the clients.
    - Providing login requests to the clients.
    - Providing IP address to clients on proper request.
    - Registering the clients.
    - Manipulating data according to the request of clients.
    - Maintaining log of the server.

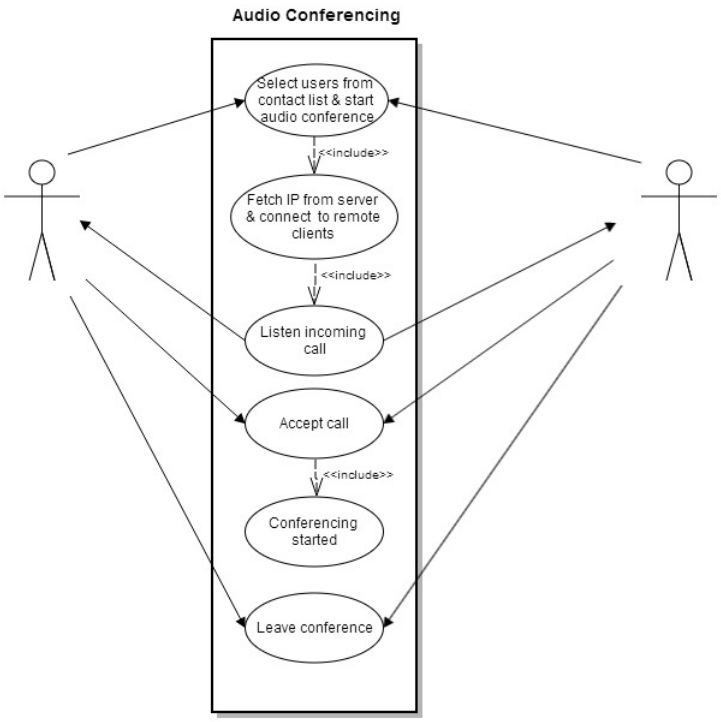


- Audio Call between two users
  - Purpose: to provide cheap facility of audio call between two users
    - First the caller goes through the contact list to select a person to call and then holds on it to find the option of calling
    - The server checks if the user is available or not, if not it sends the message to the caller that the user is not available, and if available it connects the call
    - As soon as the call connects the receiver receives a pop up window showing an incoming call which has two options : Accept and Reject.
    - The receiver can accept the call by choosing the accept option and the call will start and the two persons can communicate with each other.
    - The receiver can also reject the call If he choses reject option.
    - If the receiver does not receive the call, a missed call alert is shown.
    - Once the call is accepted the receiver or the caller both have the option to end the call, in which cases the communication between them will be put to an end.

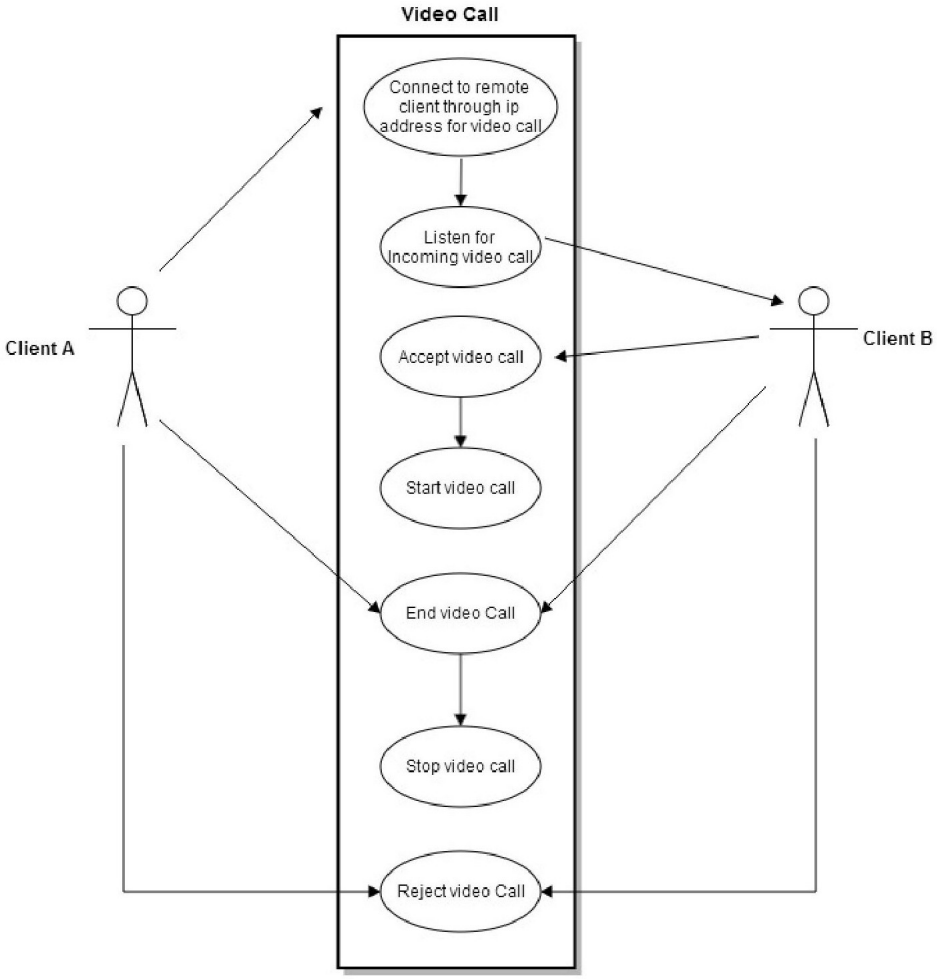


- Audio call between multiple users
  - Purpose: to provide cheap communication between many users who are within the range of Wi-Fi.
    - The sequence and stimulus is same as that of the two client user call expect that the new users are added by the call initiator.

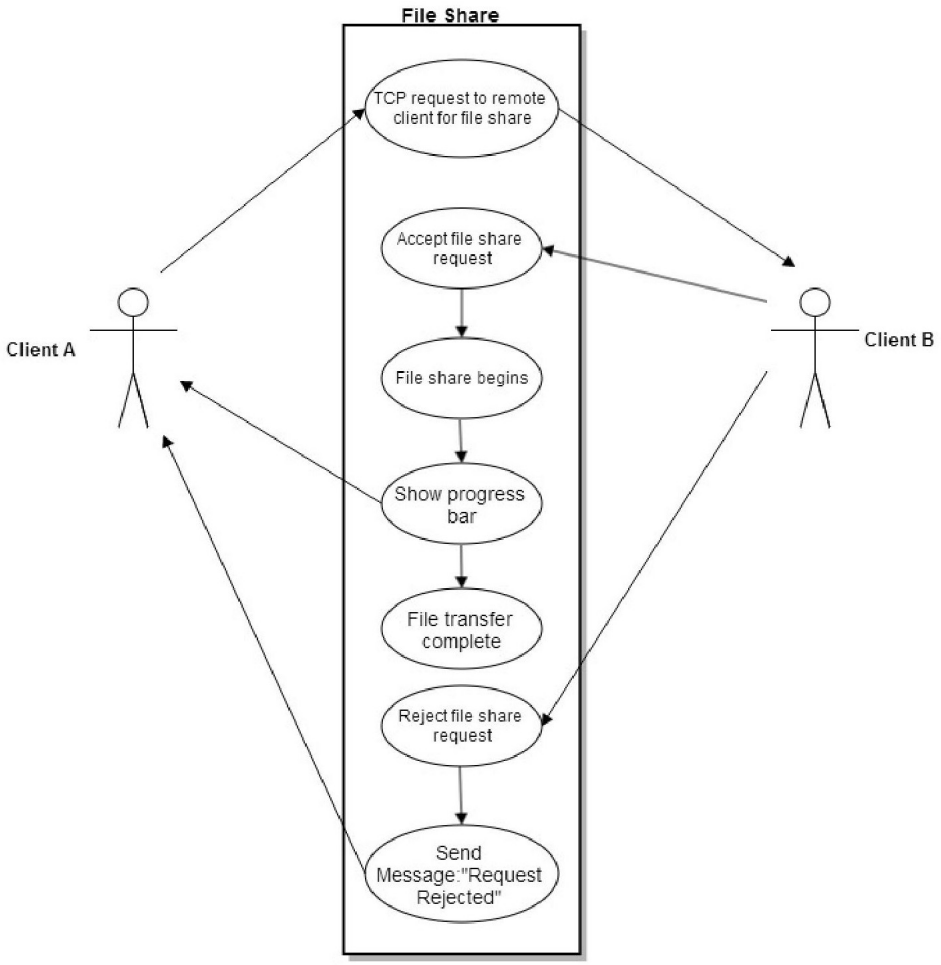
* if one user ends the call ,it doesnt affect the other users who are in the call.



- Video call between two users
  - Purpose: to provide cheap video calling between two users who can be connected via a Wi-Fi. Sequence events
    * Client can start the video call via the call button, if the other client is available the video call is connected and video streams are sent and received at both ends.
    * Client can disconnect the call by the stop button.
    * Other functionalities are same as that of the audio call.



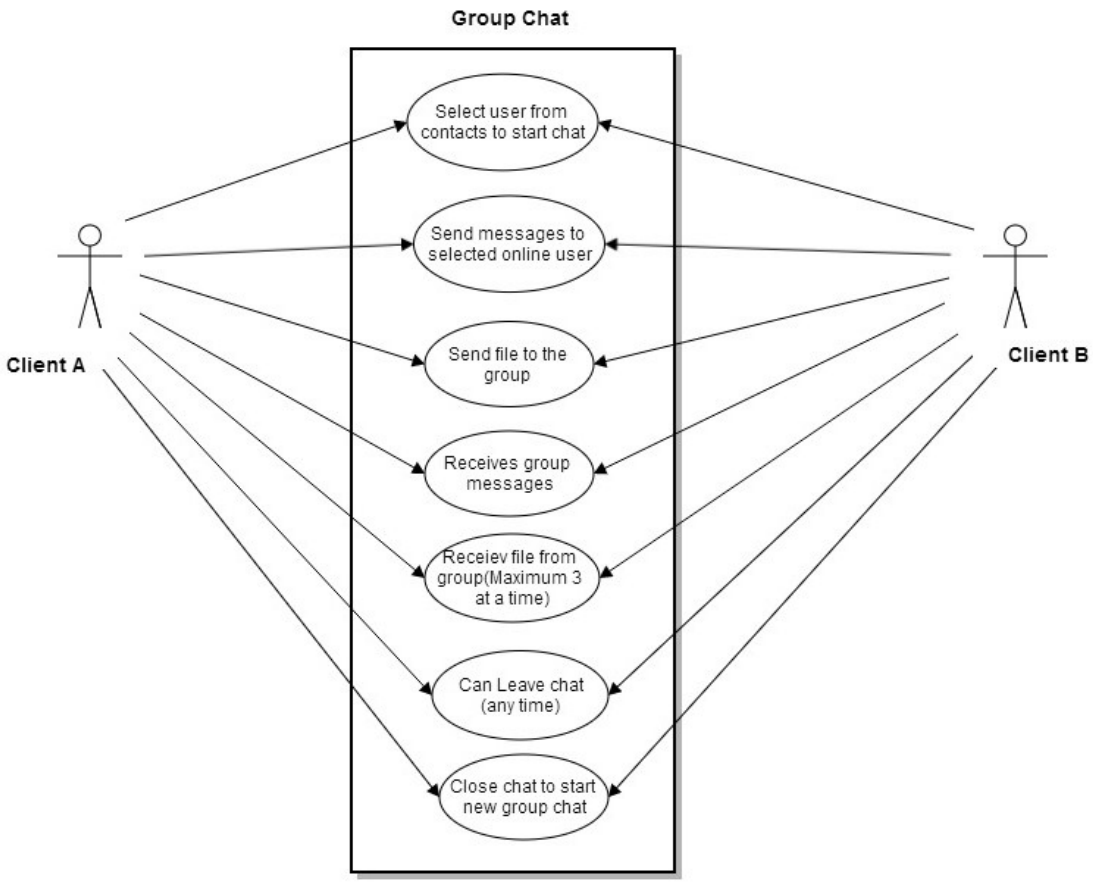
- File transfer from one user to another
  - Purpose: to send files as attachments which are present in the SD card of the sender to receiver which is available on the connected Wi-Fi.
    - The sender selects the file to be sent via a browse button, selects the contact to which the file must be sent and sends the file .
    - The file is sent only if the other user is available and is logged in.
    - If the file is sent successfully ,a toast is shown at the sender side that the file was successfully sent.
    - If the file couldnt be sent to the other user, it shows the toast that the other user is offline.



  - ∗ When the file is received at the receiver side ,the receiver sees a toast that a particular file ,from a particular sender is sent to the receiver.

- Group Chat application with file sharing.
  - Purpose: To facilitate easy transfer of text and file among multiple users who are within the range of wireless connectivity within same network.
    - ∗ The user selects multiple/single contact from his contact list and starts the Group Chat.
    - ∗ A Group Chat is started by checking the list of online users from the server.
    - ∗ A notification is sent to all selected online contacts about the Group chat.
    - ∗ Each user can now chat with every other user in the Chat room.

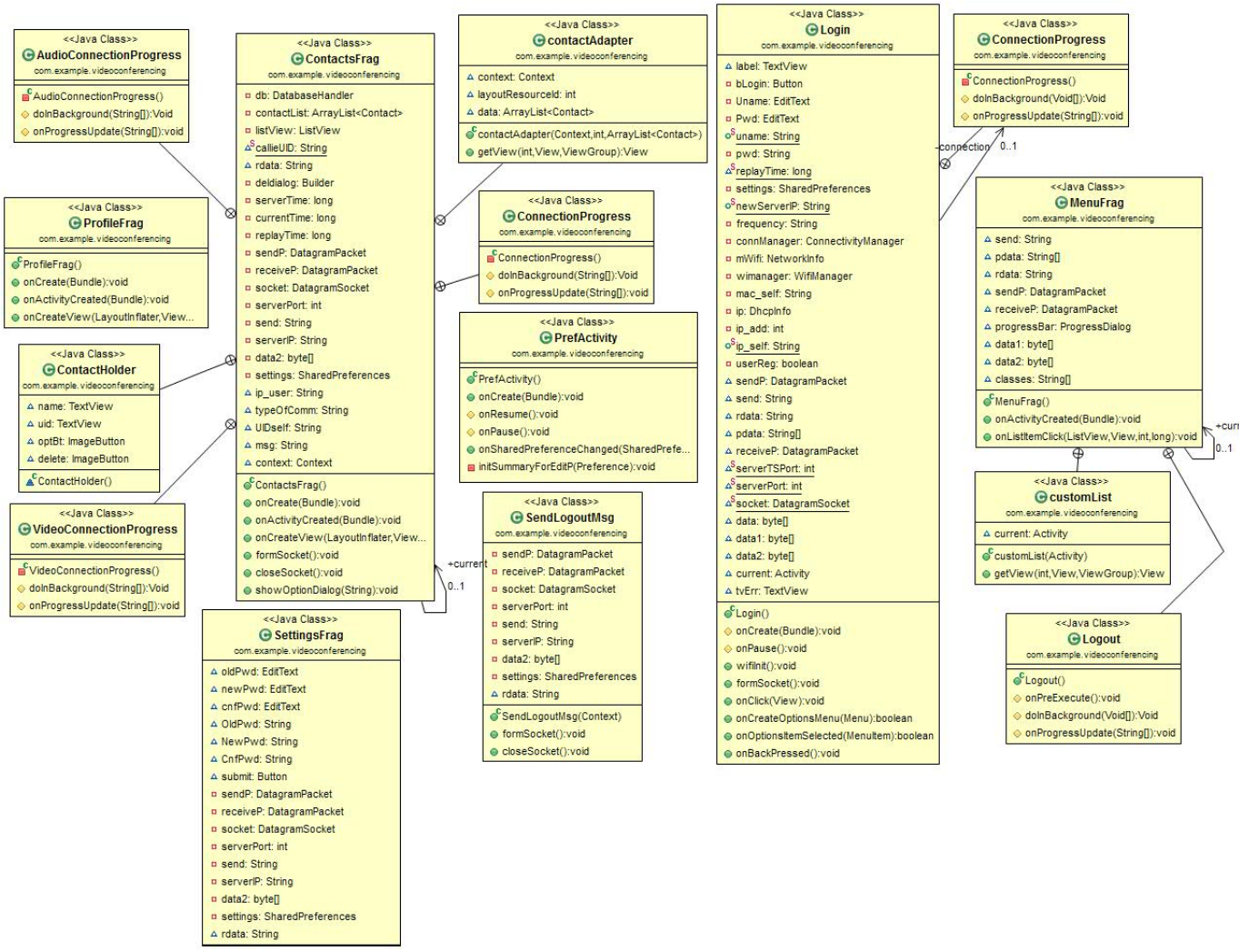
∗ A file can be selected by clicking the Browse button and can be shared in the chat room.



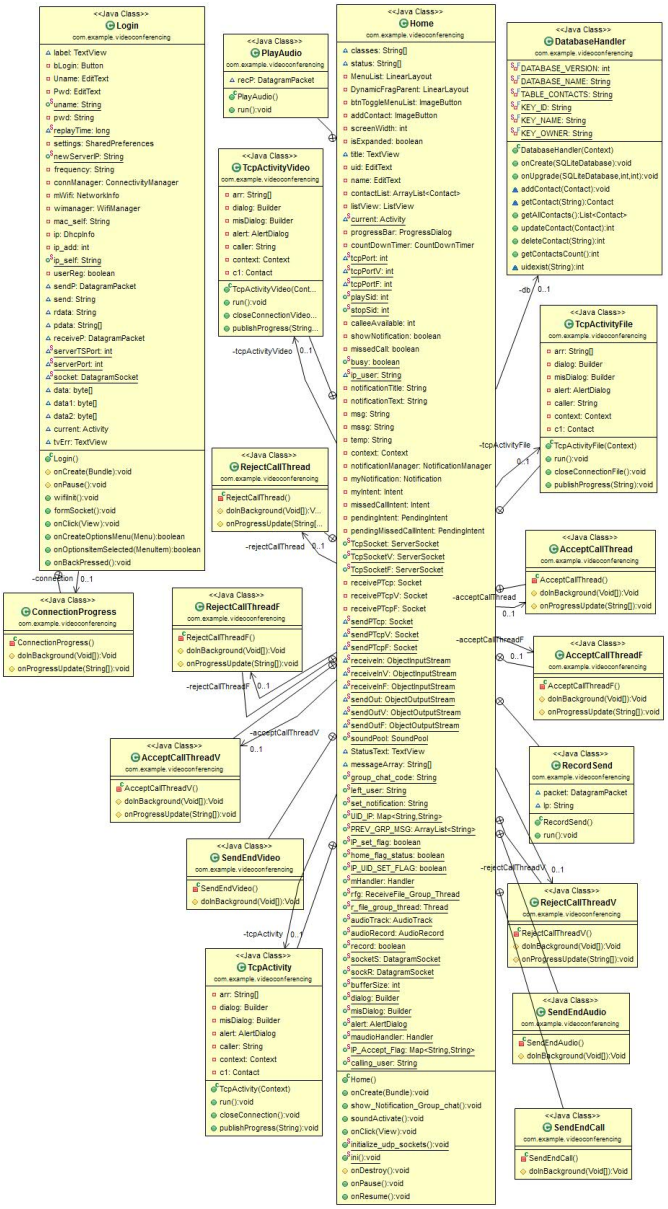
Class Diagram

The following diagrams shows different the user defined classes functional in our application:

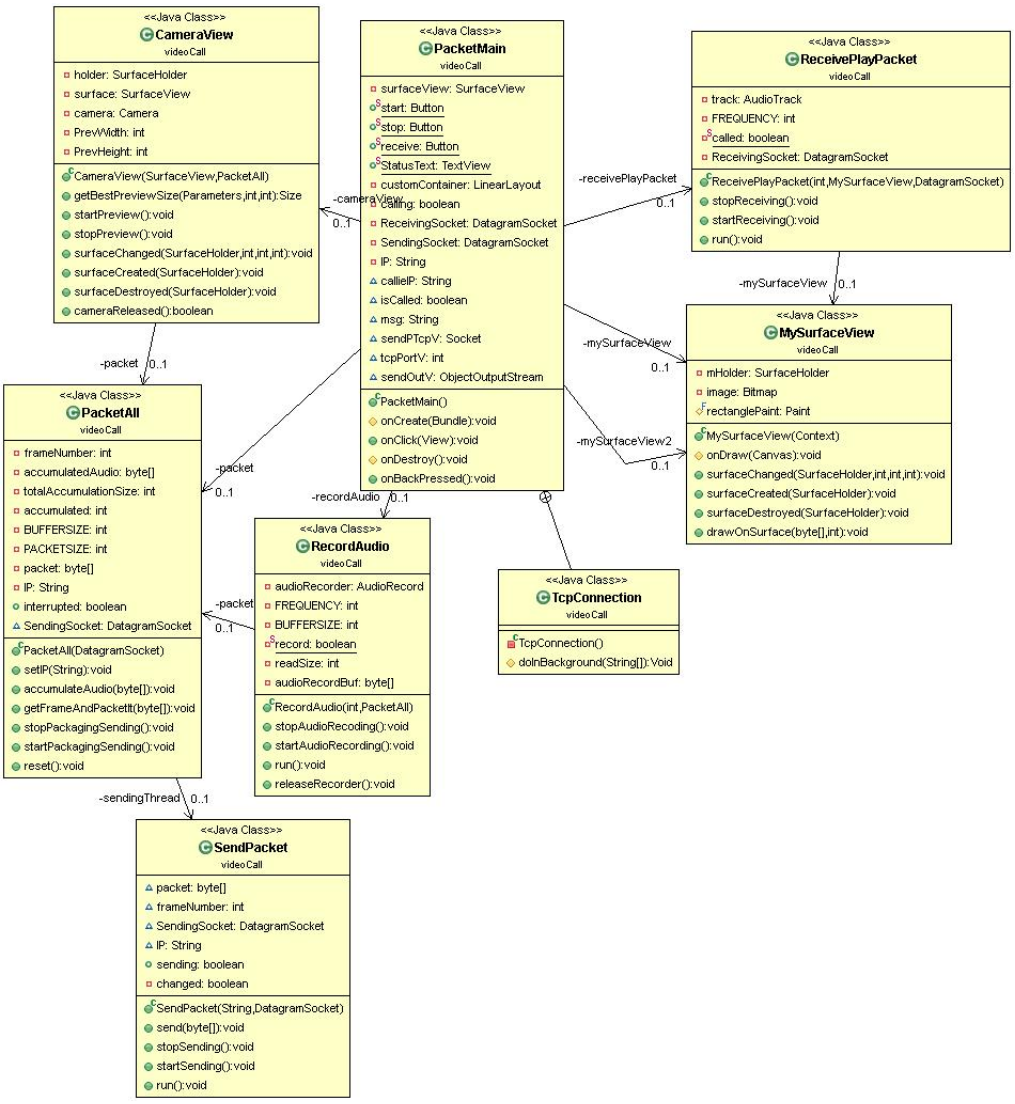
Login Class


<<Java Class>>
AudioConnectionProgress
com.example.videoconferencing
AudioConnectionProgress()
doInBackground(String[]):Void
onProgressUpdate(String[]):void
<<Java Class>>
ContactsFrag
com.example.videoconferencing
db: DatabaseHandler
contactList: ArrayList<Contact>
listView: ListView
callieUID: String
rdata: String
deldialog: Builder
serverTime: long
currentTime: long
replayTime: long
sendP: DatagramPacket
receiveP: DatagramPacket
socket: DatagramSocket
serverPort: int
send: String
serverIP: String
data2: byte[]
settings: SharedPreferences
ip_user: String
typeOfComm: String
UIDself: String
msg: String
context: Context
ContactsFrag()
onCreate(Bundle):void
onActivityCreated(Bundle):void
onCreateView(LayoutInflater,View...
formSocket():void
closeSocket():void
showOptionDialog(String):void
<<Java Class>>
contactAdapter
com.example.videoconferencing
context: Context
layoutResourceId: int
data: ArrayList<Contact>
contactAdapter(Context,int,ArrayList<Contact>)
getView(int,View,ViewGroup):View
<<Java Class>>
Login
com.example.videoconferencing
label: TextView
bLogin: Button
Uname: EditText
Pwd: EditText
uname: String
pwd: String
replayTime: long
settings: SharedPreferences
newServerIP: String
frequency: String
connManager: ConnectivityManager
mWifi: NetworkInfo
wimanager: WifiManager
mac_self: String
ip: DhcpInfo
ip_add: int
ip_self: String
userReg: boolean
sendP: DatagramPacket
send: String
rdata: String
pdata: String[]
receiveP: DatagramPacket
serverTSPort: int
serverPort: int
socket: DatagramSocket
data: byte[]
data1: byte[]
data2: byte[]
current: Activity
tvErr: TextView
Login()
onCreate(Bundle):void
onPause():void
wifiInit():void
formSocket():void
onClick(View):void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
onBackPressed():void
<<Java Class>>
ConnectionProgress
com.example.videoconferencing
ConnectionProgress()
doInBackground(Void[]):Void
onProgressUpdate(String[]):void
connection
0..1
<<Java Class>>
ProfileFrag
com.example.videoconferencing
ProfileFrag()
onCreate(Bundle):void
onActivityCreated(Bundle):void
onCreateView(LayoutInflater,View...
<<Java Class>>
ConnectionProgress
com.example.videoconferencing
ConnectionProgress()
doInBackground(String[]):Void
onProgressUpdate(String[]):void
<<Java Class>>
PrefActivity
com.example.videoconferencing
PrefActivity()
onCreate(Bundle):void
onResume():void
onPause():void
onSharedPreferenceChanged(SharedPrefe...
initSummaryForEditP(Preference):void
<<Java Class>>
MenuFrag
com.example.videoconferencing
send: String
pdata: String[]
rdata: String
sendP: DatagramPacket
receiveP: DatagramPacket
progressBar: ProgressDialog
data1: byte[]
data2: byte[]
classes: String[]
MenuFrag()
onActivityCreated(Bundle):void
onListItemClick(ListView,View,int,long):void
<<Java Class>>
ContactHolder
com.example.videoconferencing
name: TextView
uid: TextView
optBt: ImageButton
delete: ImageButton
ContactHolder()
<<Java Class>>
VideoConnectionProgress
com.example.videoconferencing
VideoConnectionProgress()
doInBackground(String[]):Void
onProgressUpdate(String[]):void
<<Java Class>>
SendLogoutMsg
com.example.videoconferencing
sendP: DatagramPacket
receiveP: DatagramPacket
socket: DatagramSocket
serverPort: int
send: String
serverIP: String
data2: byte[]
settings: SharedPreferences
rdata: String
SendLogoutMsg(Context)
formSocket():void
closeSocket():void
+current
0..1
<<Java Class>>
customList
com.example.videoconferencing
current: Activity
customList(Activity)
getView(int,View,ViewGroup):View
<<Java Class>>
SettingsFrag
com.example.videoconferencing
oldPwd: EditText
newPwd: EditText
cnfPwd: EditText
OldPwd: String
NewPwd: String
CnfPwd: String
submit: Button
sendP: DatagramPacket
receiveP: DatagramPacket
socket: DatagramSocket
serverPort: int
send: String
serverIP: String
data2: byte[]
settings: SharedPreferences
rdata: String
<<Java Class>>
Logout
com.example.videoconferencing
Logout()
onPreExecute():void
doInBackground(Void[]):Void
onProgressUpdate(String[]):void

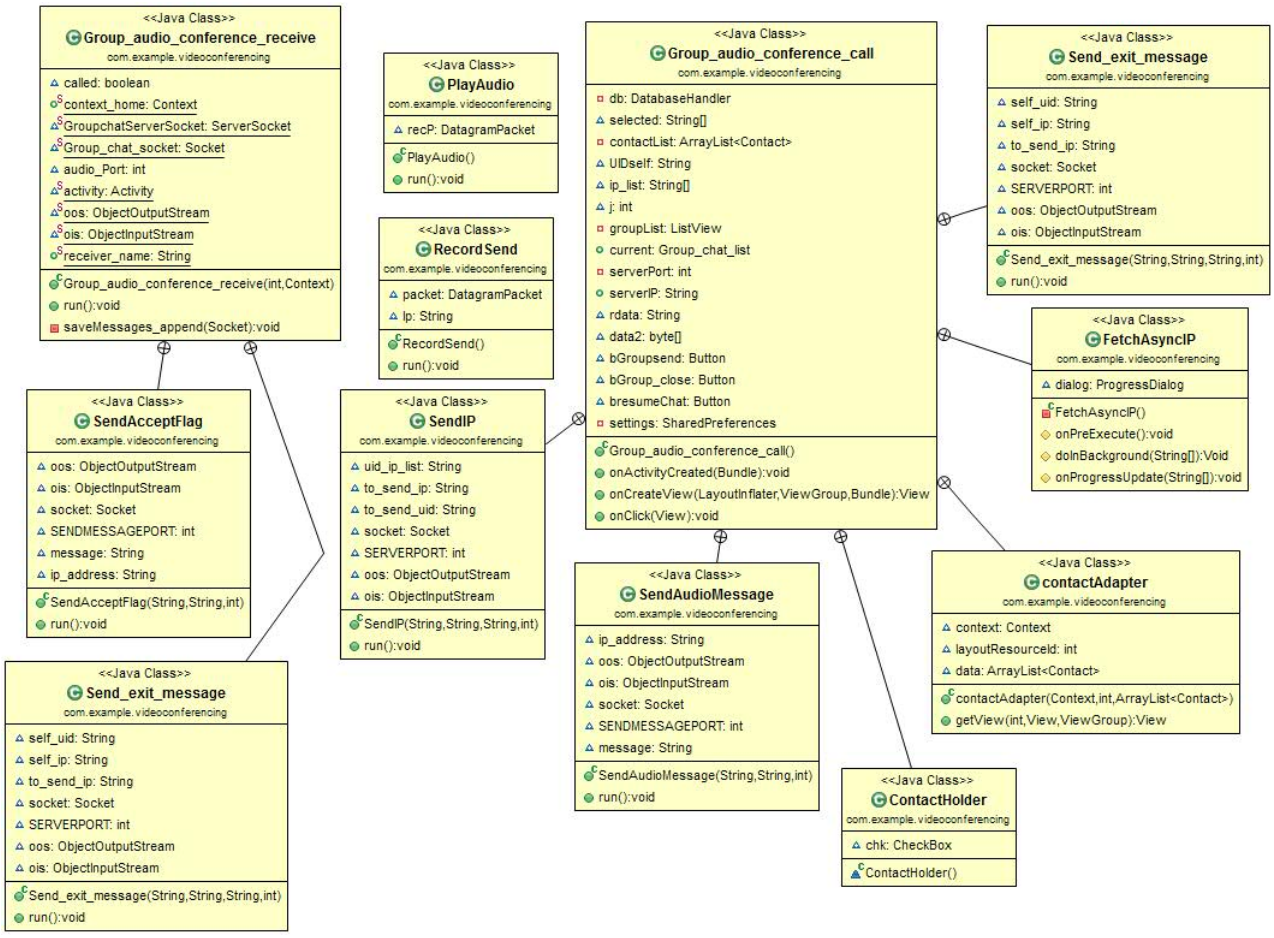
Home Class


Login
com.example.videoconferencing
label: TextView
bLogin: Button
Uname: EditText
Pwd: EditText
uname: String
pwd: String
replayTime: long
settings: SharedPreferences
newServerIP: String
frequency: String
connManager: ConnectivityManager
mWifi: NetworkInfo
wimanager: WifiManager
mac_self: String
ip: DhcpInfo
ip_add: int
ip_self: String
userReg: boolean
sendP: DatagramPacket
send: String
rdata: String
pdata: String[]
receiveP: DatagramPacket
serverTSPort: int
serverPort: int
socket: DatagramSocket
data: byte[]
data1: byte[]
data2: byte[]
current: Activity
tvErr: TextView
Login()
onCreate(Bundle):void
onPause():void
wifiInit():void
formSocket():void
onClick(View):void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
onBackPressed():void
PlayAudio
com.example.videoconferencing
recP: DatagramPacket
PlayAudio()
run():void
TcpActivityVideo
com.example.videoconferencing
arr: String[]
dialog: Builder
misDialog: Builder
alert: AlertDialog
caller: String
context: Context
c1: Contact
TcpActivityVideo(Cont...
run():void
closeConnectionVideo...
publishProgress(String...
-tcpActivityVideo
0..1
RejectCallThread
com.example.videoconferencing
RejectCallThread()
doInBackground(Void[]):V...
onProgressUpdate(String[...
-rejectCallThread
0..1
-connection
0..1
ConnectionProgress
com.example.videoconferencing
ConnectionProgress()
doInBackground(Void[]):Void
onProgressUpdate(String[]):void
RejectCallThreadF
com.example.videoconferencing
RejectCallThreadF()
doInBackground(Void[]):Void
onProgressUpdate(String[]):void
-rejectCallThreadF
0..1
-acceptCallThreadV
0..1
AcceptCallThreadV
com.example.videoconferencing
AcceptCallThreadV()
doInBackground(Void[]):Void
onProgressUpdate(String[]):void
SendEndVideo
com.example.videoconferencing
SendEndVideo()
doInBackground(Void[]):Void
-tcpActivity
0..1
TcpActivity
com.example.videoconferencing
arr: String[]
dialog: Builder
misDialog: Builder
alert: AlertDialog
caller: String
context: Context
c1: Contact
TcpActivity(Context)
run():void
closeConnection():void
publishProgress(String):void
Home
com.example.videoconferencing
classes: String[]
status: String[]
MenuList: LinearLayout
DynamicFragParent: LinearLayout
btnToggleMenuList: ImageButton
addContact: ImageButton
screenWidth: int
isExpanded: boolean
title: TextView
uid: EditText
name: EditText
contactList: ArrayList<Contact>
listView: ListView
current: Activity
progressBar: ProgressDialog
countDownTimer: CountDownTimer
tcpPort: int
tcpPortV: int
tcpPortF: int
playSid: int
stopSid: int
calleeAvailable: int
showNotification: boolean
missedCall: boolean
busy: boolean
ip_user: String
notificationTitle: String
notificationText: String
msg: String
mssg: String
temp: String
context: Context
notificationManager: NotificationManager
myNotification: Notification
myIntent: Intent
missedCallIntent: Intent
pendingIntent: PendingIntent
pendingMissedCallIntent: PendingIntent
TcpSocket: ServerSocket
TcpSocketV: ServerSocket
TcpSocketF: ServerSocket
receivePTcp: Socket
receivePTcpV: Socket
receivePTcpF: Socket
sendPTcp: Socket
sendPTcpV: Socket
sendPTcpF: Socket
receiveIn: ObjectInputStream
receiveInV: ObjectInputStream
receiveInF: ObjectInputStream
sendOut: ObjectOutputStream
sendOutV: ObjectOutputStream
sendOutF: ObjectOutputStream
soundPool: SoundPool
StatusText: TextView
messageArray: String[]
group_chat_code: String
left_user: String
set_notification: String
UID_IP: Map<String,String>
PREV_GRP_MSG: ArrayList<String>
IP_set_flag: boolean
home_flag_status: boolean
IP_UID_SET_FLAG: boolean
mHandler: Handler
rTg: ReceiveFile_Group_Thread
r_file_group_thread: Thread
audioTrack: AudioTrack
audioRecord: AudioRecord
record: boolean
socketS: DatagramSocket
sockR: DatagramSocket
bufferSize: int
dialog: Builder
misDialog: Builder
alert: AlertDialog
maudioHandler: Handler
IP_Accept_Flag: Map<String,String>
calling_user: String
Home()
onCreate(Bundle):void
show_Notification_Group_chat():void
soundActivate():void
onClick(View):void
initialize_udp_sockets():void
init():void
onDestroy():void
onPause():void
onResume():void
DatabaseHandler
com.example.videoconferencing
DATABASE_VERSION: int
DATABASE_NAME: String
TABLE_CONTACTS: String
KEY_ID: String
KEY_NAME: String
KEY_OWNER: String
DatabaseHandler(Context)
onCreate(SQLiteDatabase):void
onUpgrade(SQLiteDatabase,int,int):void
addContact(Contact):void
getContact(String):Contact
getAllContacts():List<Contact>
updateContact(Contact):int
deleteContact(String):int
getContactsCount():int
uidexist(String):int
-db
0..1
TcpActivityFile
com.example.videoconferencing
arr: String[]
dialog: Builder
misDialog: Builder
alert: AlertDialog
caller: String
context: Context
c1: Contact
TcpActivityFile(Context)
run():void
closeConnectionFile():void
publishProgress(String):void
-tcpActivityFile
0..1
AcceptCallThread
com.example.videoconferencing
AcceptCallThread()
doInBackground(Void[]):Void
onProgressUpdate(String[]):void
-acceptCallThread
0..1
AcceptCallThreadF
com.example.videoconferencing
AcceptCallThreadF()
doInBackground(Void[]):Void
onProgressUpdate(String[]):void
-acceptCallThreadF
0..1
RecordSend
com.example.videoconferencing
packet: DatagramPacket
ip: String
RecordSend()
run():void
RejectCallThreadV
com.example.videoconferencing
RejectCallThreadV()
doInBackground(Void[]):Void
onProgressUpdate(String[]):void
-rejectCallThreadV
0..1
SendEndAudio
com.example.videoconferencing
SendEndAudio()
doInBackground(Void[]):Void
SendEndCall
com.example.videoconferencing
SendEndCall()
doInBackground(Void[]):Void

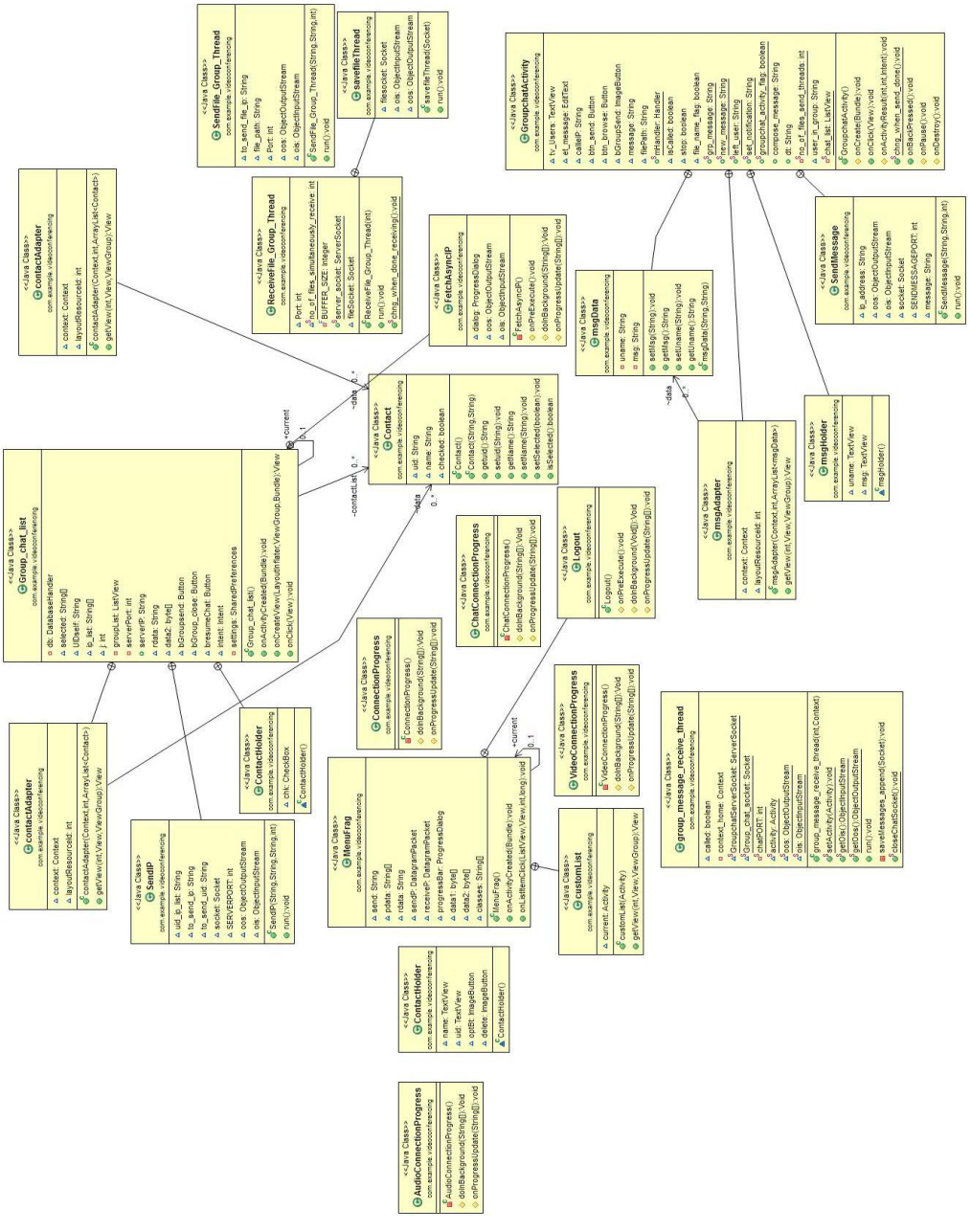
Video Call Class



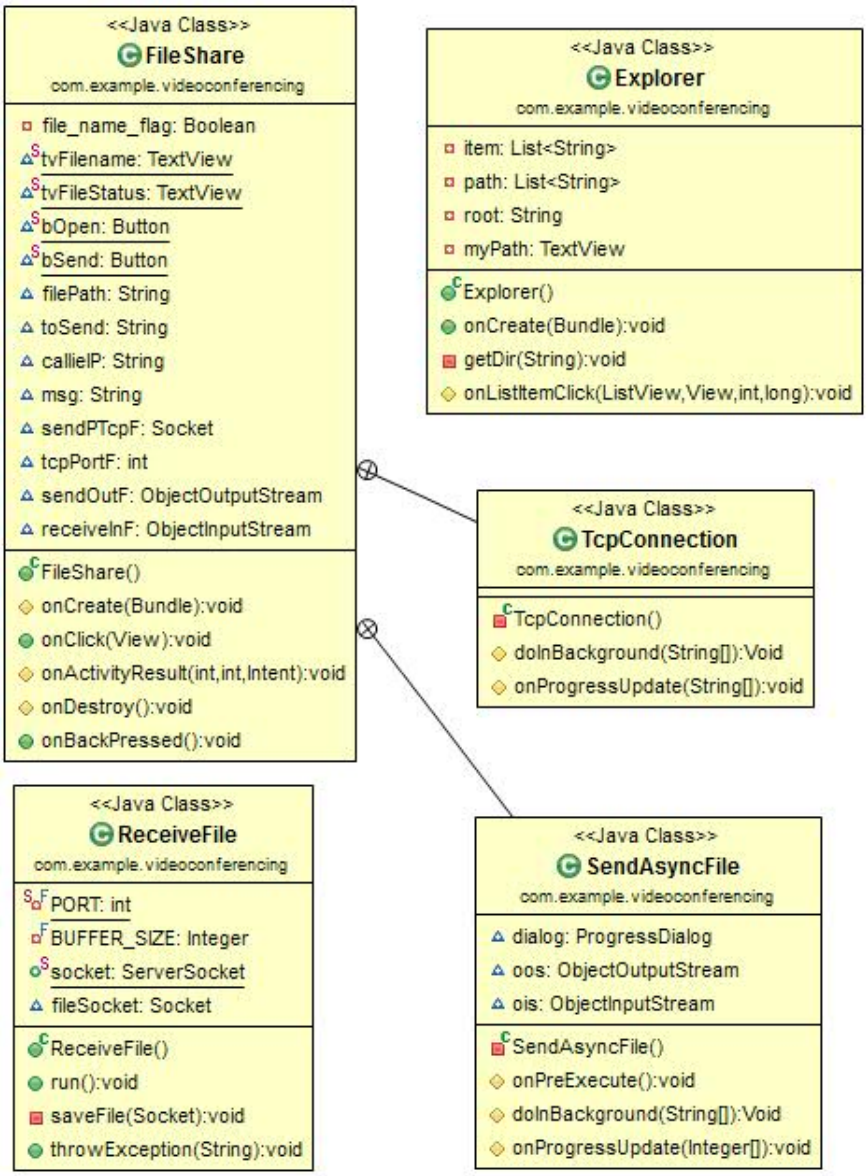
Audio Conferencing Class



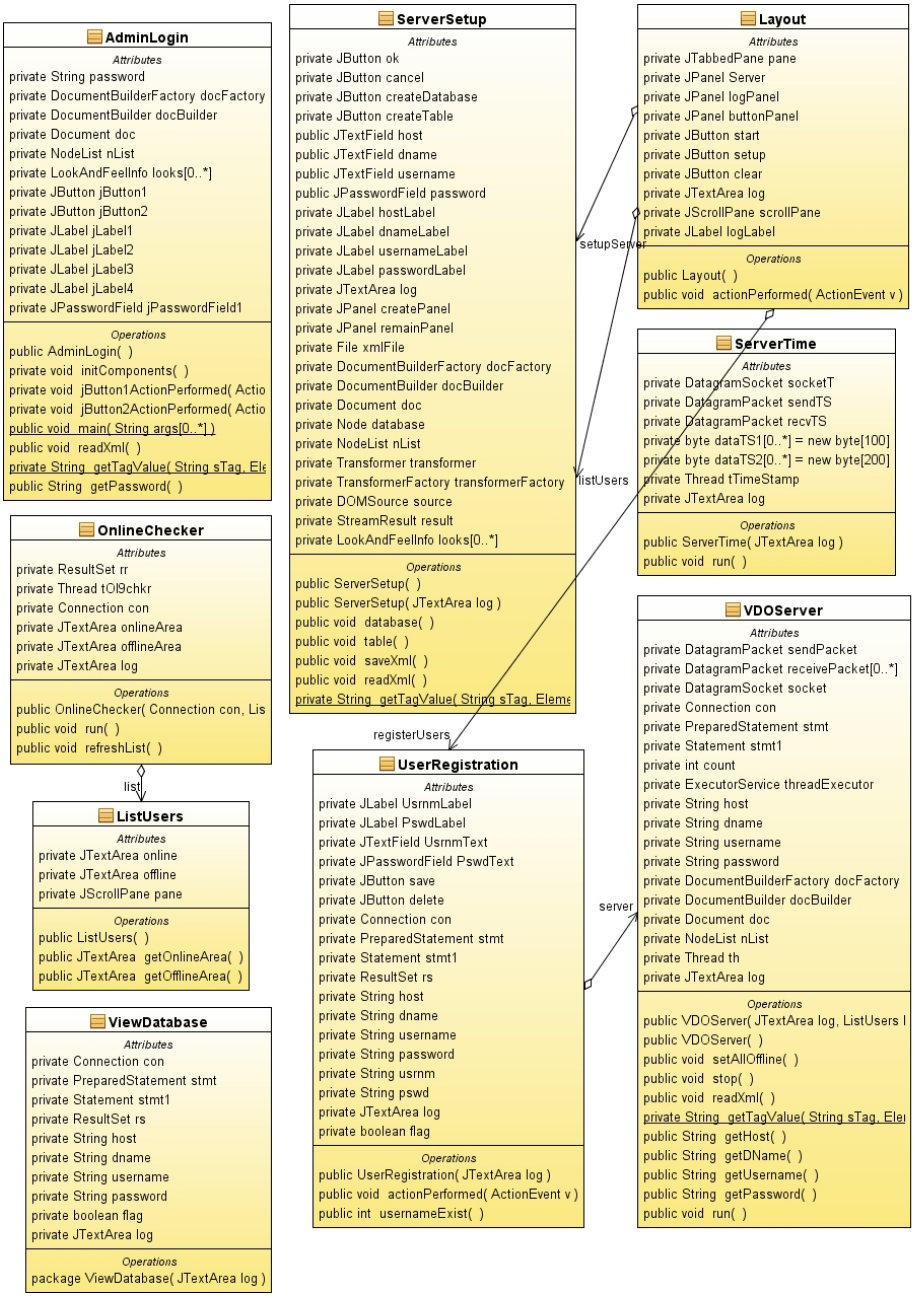
Group Chat Class



File Sharing Class



Server Class



Functions of different Classes

1. CLIENT:

- **DataBaseHandler :** This class handles the client side database. When first time application is installed, it creates a database contactManager with three fields uid, name and owner. Primary key is uid+owner. Database stores all the contact saved on the device. This class handles the operations like add contact, delete contact in the contact list.

- **TcpActivityAudio :** This thread listens for incoming call requests from all the users registered with server. When user receives a call It shows a alert box with two options accept or reject. It creates a tcp connection with the caller. And it sends and receives all the control messages during the call. This tcp connection closes when either of side ends the call.

- **TcpActivityFile :** This thread listens for incoming fil transfer requests from all the users registered with server. When user receives a request It shows a alert box with two options accept or reject. It creates a tcp connection with the caller. When user accept or reject the call, this tcp connection is closed. Receive file Thread receives the file in background.

- **TcpActivityVideo :** This thread listens for incoming video call requests from all the users registered with server. When user receives a video call It shows a alert box with two options accept or reject. It creates a tcp connection with the caller. And it sends and receives all the control messages during the call. This tcp connection closes when either of side ends the call.

- **AccepCallThread :** This thread starts working when user accepts the audio call.

- **RejectCallThread :** This thread starts working when user rejects the video call.

- **AccepCallThreadF :** This thread starts working when user accepts the file transfer request.

- **RejectCallThreadF :** This thread starts working when user rejects the file transfer request.
- **AccepCallThreadV :** This thread starts working when user accepts the video call request.
- **RejectCallThreadV :** This thread starts working when user rejects the video call request.
- **PlayAudio :** This thread starts working when a audio call starts. It plays the UDP audio packets received from remote user.
- **RecordSend :** This thread starts when a audio call starts, It forms UDP audio packets and send them to remote users.
- **TcpConnection(Audio) :** This thread makes a TCP connection with remote user when user makes audio call.
- **Contact :** This class has two fields uid and name. Object of this class represent the contact.
- **ContactsFrag :** This class maintains all the contact activities.
  When user select a contact for audio/video/file this class initiates the threads AudioConnection/VideoConnection/ConnectionProgress.
- **AudioConnectionProgress :** This thread starts working, when user selects a contact for audio call. It shows the message like "user is not online","contact is not registered on server" or "user is busy". If user is online it receives the IP of user from server and makes audio call.
- **VideoConnectionProgress :** This thread starts working, when user selects a contact for video call. It shows the message like "user is not online","contact is not registered on server" or "user is busy". If user is online it receives the IP of user from server and makes video call.
- **ConnectionProgress :** This thread starts working, when user selects a contact for file transfer. It shows the message like "user is not online","contact is not registered on server" or "user is busy". If user is online it receives the IP of user from server and makes file transfer request to remote user.

- **SettingsFrag :** This class is used to change password. User have to provide three things: Old password, new password and confirm passwod.
- **FileShare :** This class starts working when user makes a file transfer request to remote user.
- **TcpConnection(File) :** It establishes a TCP connection with remote user to make a file transfer request.
- **SendAsynFile :** This background process starts when user send a file to remote user.It shows a dialogue box to show the amount of file transferred.
- **Login :** This class loads when user starts the app. It fetches the user preferences(server IP) saved on userthe device and send the login request to the server. On succesful login Home intent is opened.
- **PrefActivity :** This class save the preferences of user on the device. User can give his preference on the login page.
- **Group_audio_conference_call :** This class handles the main functionality of retrieving Ip's of selected users(for audio conference) from the server and sending them appropriate messages on clicking "start conference" and "stop conference" button.
- **SendIp :** This class sends the List of Ip's of all the users selected in the audio conference, to each and every user in the conference. This class starts when conference initiator starts the conference.
- **SendAudioMessage :** This class is used to send audio conference starting request to the selected users from audio conferencing page.
- **Send_exit_message :** This class is used to send exit message to all the users in the conference call. This class is invoked when any user leaves the conference.
- **ContactAdapter :** This class creates a view for displaying the contacts of the user with a checkbox(for selecting them for audio conference).
- **FetchAsyncIp :** This class fetches Ip's of selected users(for audio conference) from the server.

- **Group_audio_conference_receive :** This class handles the functionality of receiving audio conference requests from other users and also actions to be taken after receiving different types of requests.
- **SendAcceptFlag :** This class sends a confirmation to all users in the conference call that he she has accepted the call request and is now starting communication.
- **saveMessages_append :** This class decodes messages received from other users and performs actions according to received messages.
- **Send_exit_message :** This class is used to send message to all other users in the conference call that he she has rejected the conference request.
- **Group_chat_list :** This class extends the fragment class, which is used to generate the list of friends added by the user in the contats in the selected fragment of group chat. Also the implementation of this class starts the group chat between all the selected friends from the contact list by fetching the IP address from the server.
- **GroupchatActivity :** This is the main UI class for having a group chat, this class has its own layout for showing messages, the list for showing online users and selecting the file and sending the same to the whole group. The messages and file are sent in seperate threads to all the selected friends.
- **SendMessage :** Inner class of group chat activity to send messages to the friends selected in the group chat.
- **group_message_receive_thread :** A seperate thread which starts at the home page, the thread handles all the messages requests on a particular port, either it is a new group chat request or a new message or an exit message. This thread also replies for a bad request.
- **SendFile_Group_Thread :** A thread class which spawn seperate threads and manages all the spawned threads for sending, completion, failures of the specified file to all the friends in the group simulataneously.
- **ReceiveFile_Group_Thread :** A thread class which spawn seperate threads and manages all the spawned threads for validating file request with group

code start receiving, completion, failures of the file from the sending party, a maximum of 3 simultaneously receive is possible.

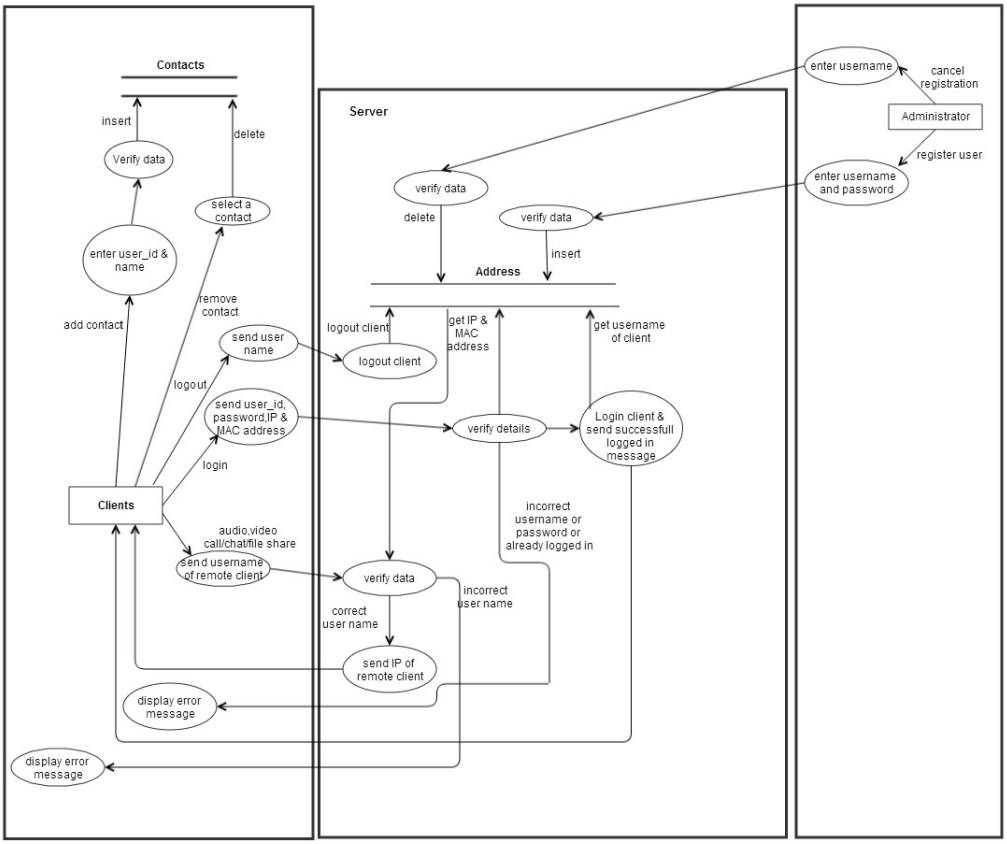
- **Send_exit_message :** A thread class to inform all the friends in selected group that the he has left the chat.

2. SERVER:

- **VDOServer:** It initiates the OnlineChecker and ServerTime threads. It continuously listens at port 6500 for all client requests. It creates a new ProcessPacket thread to process each received packet from a client.
- **OnlineChecker:**This class pings each online client every 5 minutes. If it does not receive a reply it updates the database to mark the client as offline.
- **ServerTime:** Send the current time of the server to the requesting clients.
- **ServerSetup:** sets up the mysql server and connects to it by taking the details from admin
- **Layout:**consists the main server frame
- **ListUsers:** shows the present number of online and offline users with their names
- **UserRegistration:** registers a user, deletes him from the database and forcefully kicks him out of the server when required

2.3.2 Data Flow Diagram

- Video Conferencing Application



2.3.3 Low Level Design Document

Algorithm & Flowchart

- Audio Call
  - Client End:

    1 . The Client either click on the name of the person in his/her own contact list and if the client to be called is not in the contact list of the call-initiator client, he can add the person to his contact list. a) Shehe needs to know the unique id of the other person, he/she wants to add in his contact list and if the unique_id is found by the server at the server side, the server sends a message to the client that the other user is not registered, otherwise adds this user to the contact of the caller client.

2 . The server returns the ip_add of the person who needs to get called, this ip_address is returned to the client who wants to initiate the call and the peer to peer then gets started.

3 . Only clients who are registered at the server side can be called (with their unique_id) .

4 . Create 2 TCP sockets (to communicate with another online client)

5 . If the user wants to make a call, he has to click on the start button and if the user wants to end a ongoing call, click on the stop button. If the other user is not online, a toast is shown at the client side that the other client is not online.

6 . Else, for an Incoming Call, ringing gets started on the other side and a toast is shown, showing the name of the person who is calling.

a . If "accept",

i . Start call

ii . End call

b . Else if "reject", reject call.

7 . If the client on the other side has rejected the call of the sendercaller client, a toast is shown at hisher side that the user has rejected the call and the toast which till now showed -connecting... now disappears.

8 . The audio packets are sent through the UDP protocol.

9 . The ringing and the call connecting process is handled by the TCP mechanism.

10 . Both the callers need to use the earphone to have better quality.

– Video Calling: Assuming that both client A and B are logged in:

1 . A TCP request is forwarded by Client A to B, if Client B is not busy it accepts the request.

2 . A pop up window appears on Client Bs screen for accepting or rejecting the video call.

3 . If Client B accepts the call, Client A is notified to start sending and receiving UDP packets to Client B and B also starts sending and receiving packets simultaneously.

4 . The sending thread on both the clients prepare and extract the packets to be sent in following way.

- Camera sends frames, which is compressed to JPEG for memory consumption to the thread one by one which is combined with the audio recorded in the buffer and a packet is formed which is sent via UDP connection as a datagram packet.
- Along with audio and video data, audio length, video length and frame numbers are also inserted in the packet.
- The receiving thread receives the packet and extracts the audio and video data in different buffers. The JPEG image is drawn on canvas and the audio is fed to track to be played.

5 . Any Client can end the call at any point of time. Then the other client is notified about the end of call.

– File Share Assuming that the client A and B are online:

1 . A receiving thread always runs in the background of activity which listens to the incoming request of file share from remote client.

2 . Client A selects a user from contacts list and selects file share option.

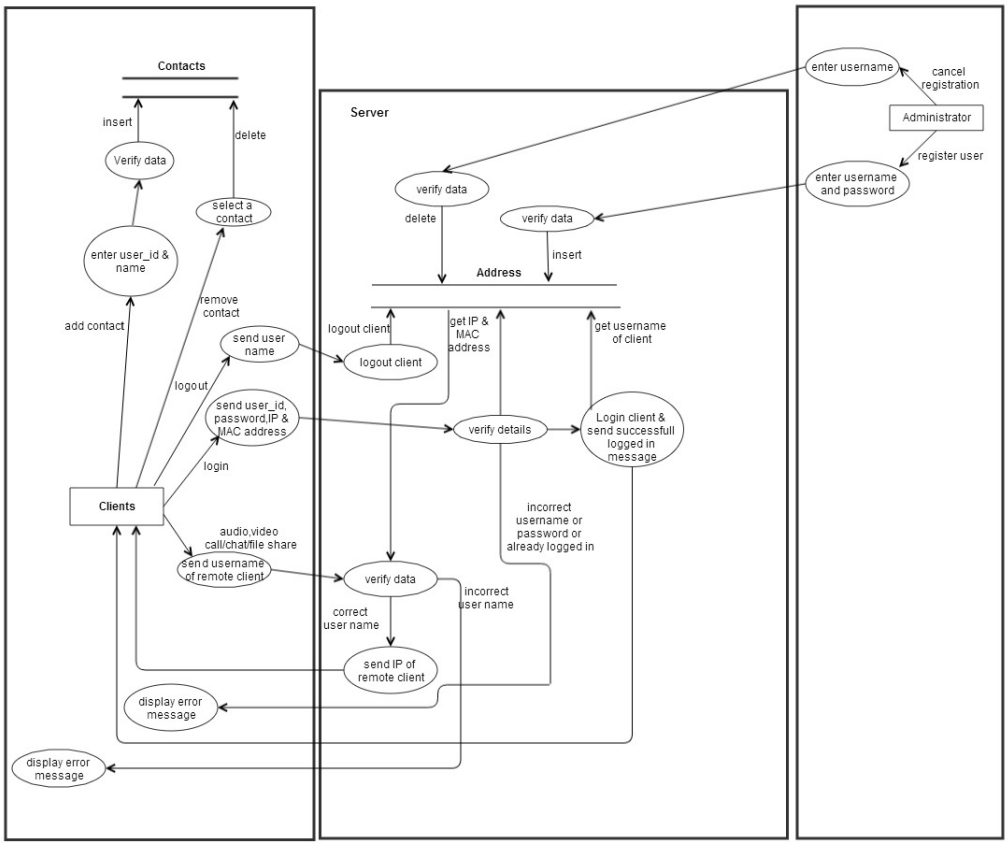
3 . A message is sent to server about the type of operation i.e. file share in this case with username and UId.

4 . The server returns the IP address corresponding to that UID.

5 . A TCP request is forwarded to remote client which opens a pop-up window on remote clients side which has accept and reject option.

- If remote client selects accept, then Client A is notified that the remote client has accepted the request.
- Client A is taken to the main thread of file share where it can select the file to be sent by OPEN button.
- It gives the list of the contents of SD card to be selected for file transfer.
- Once a file is selected, it can then be sent to the remote client by clicking on the sent button.
- If the remote user rejects the request for file share, a message is sent to Client A that its request has been rejected.

* It uses TCP protocol to send the file to remote client.
* While the file is being sent, a progress bar is shown and the file is sent in the background using



– Group Chat. Algorithm for group Chat and Group File Share

1 .A GroupMessageReceive(GMR) Thread is running which is always listening for a TCP socket connection on a specific port, when the user logs in for a message or a new group chat request.

2 .A ReceiveFile_Group_Thread Thread is running which is always listening for a TCP socket connection on a specific port, when the user logs in, for file valid file request from a group.

3 .The chat starting party starts the chat by selecting a group of contacts from his contact list and request the server for the online users and and process the response accordingly.

4 If any user is online, a chat room is opened showing the list of online user/users, sending the chat request to all the selected online user.

· Request contains:

· - who started the group chat.

· - A GroupChatCode with the userid, IP details of all the online users in the

group.

5 . Whenever a group chat request arrives, the corrosponding GMR thread in response accepts the connection it accepts a list of IP address and sets the received GroupChatCode as its own chat code, by this way every online user in the chat has the same GroupChatCode.

6 . A MAP data structure is used to store the IP address and username of each user. A groupchat_flag is set to true whenever a group chat is started so that no other group chat can be start.

7 . If the requested user is already busy in a chat he sends an exit message to the requesting party, and that request is processed.

- An Exit message is sent by the busy party to the users in the received userid IP list.
- The Exit message is interprated as left the room by all the users in the online list and message is shown as User Left the room.

8 . Now online users in the room can either receive a message through a notification(when ever he is on a homescreen), he can go to the chat room by clicking on Show chat button given in the group chat window and continue chat any time till all the user are in the room.

- If no user is in the room a toast is shown for notification.

8 . Sending of messages are done in seperate threads(done for listening to new connection request and proper real time communication in the network and avoid delay and latency) for each user in the room (For each person in the Map(Containg UID & IP), iteratively a new thread is started to send message).

- The sent messages are stored in an ArrayList which contains all the logs of messages for the session.

9 . The GMR threads listnes the message request and spawns a thread to receive the message and update the chat window andArraylist both.

10 . Group file share is also an option, to send file one at a time to all the users of that particular chat room.

- This is done by creating threads for each user in the room to send file concurrently to all users for.

11 . File receive thread filters file receive request by the group code.

- If group code matches it spawns a new thread which communicates with the sending thread to continues to write file onto the disk.
- Notifications are shown for every file receive request and the status(On Homescreen a Toast is shown).
- Message are logged in the room of user about the file and current status of file(Success or Fail(ArrayList Updated)).
- A maximum of 3 file simultaneous receive is possible, to minimize the load on devices.
- While receiving a file, also the user can send a file or chat(Network and Device dependent).
- If ChatGroupode do not matches, it ignores the request.

12 . The messages for the a group chat session is stored in an ArrayList, which is updated by the all the threads which receives a proper message request either for a chat message, file share request, Exit requests.

13 .Closes Chat or exits from the Group chat.

- The thread sends an exit message to all the users in its MAP.
- Each receiving part modifies its GroupChatCode to maintain a proper flow of data.
- The MAP is cleared which store user and IP details.
- The ArrayList containing the messages are cleared.
- The group chat flag is set to false and it starts listening new requests.
- A notification or toast saying __ user has left the chat room.

14 . Group chat fragments added on start of group chat with selected users .

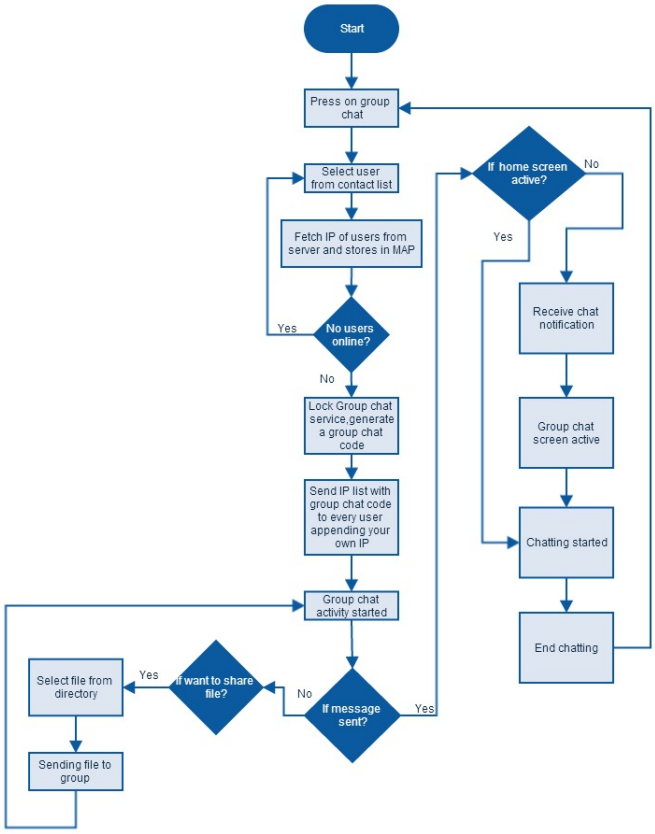
- The client side requests server for IPs of the selected users.
- When the client side receives IP address of the selected users, it saves those IP address in a MAP structure with key as username and IP as value.

15 . Now, each person is sent the group code and IP addresses of the rest of the persons in the group chat.

16 . The sending thread either sends a file or a message, and accordingly a receiving thread receives the message or file on two different ports.

- For each person in the map, iteratively a new thread is started to send message/file request.

17 . When other users receive exit message from a particular user, the IP address of that user is removed from the MAP and a new group code is generated.



- Audio Conference Call

1 . A GroupAudioCallReceive(GACR) Thread is running which is always listening for a TCP socket connection on a specific port, when the user logs in for an audio conference call request.

2 . The audio conference starting party starts the call by selecting a group of contacts from his contact list and request the server for the online users and and process the response accordingly.

3 . If any user is online, then audio call request is sent to him/her.

* Request contains:
* - Who started the Audio Conference Call.
* - A GroupAudioCode with the userid, IP details of all the online users in the group.

4 . Whenever an Audio Conference Call request arrives, the corrosponding GACR thread in response accepts the connection along with a list of IP address and sets the received GroupAudioCode as its own audio call code, by this way every online user in the chat has the same GroupAudioCode.

5 . Two MAP data structures are used to store the IP address, username (in first) and IP address, flag whose default value is set to false for all users (in second) of each user. A groupaudio_flag is set to true whenever an audio conference call is started so that no other audio call can be started.

6 . At the receiving end,

i if the user accepts the call, then the flag value in second map for corresponding user is updated to true.

* An accept message is sent to all others listed in the first map .
* RecordSend Thread and PlayAudio thread are started for recording & sending audio and receiving & playing audio respectively.

ii if the user rejects the call, then an exit message is sent to all other users whose IPs' are in first map.

* The corresponding sockets are closed, the maps are cleared and a Toast is shown that "You have Rejected the Call".

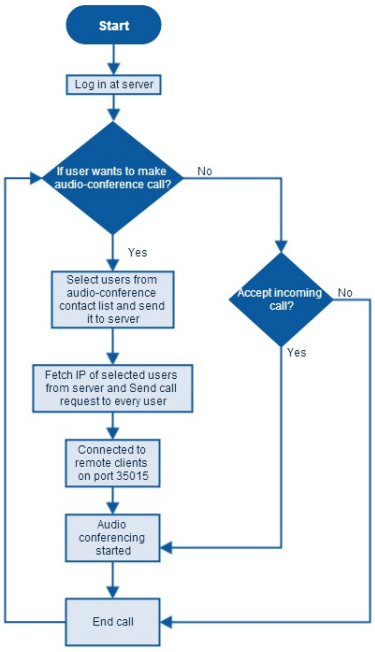
7 . At the initiator of the call, a Toast is displayed "User _ has accepted your Call", and the corresponding RecordSend & PlayAudio thread are invoked.

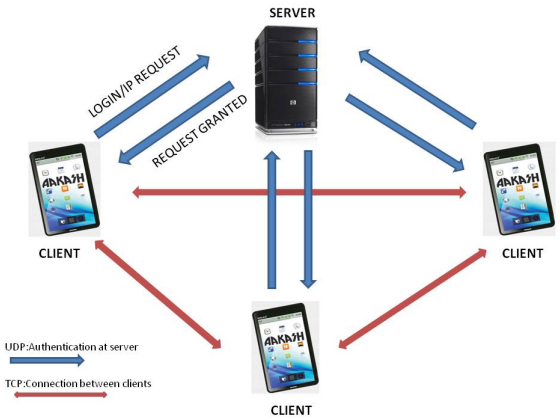
8 . The users can now talk over the Tablets.

9 . On either end, if the user leaves the audio conference call, then at first an exit message is sent to all other users in his maps whose flag value are true.

10 . Then all the sockets are closed, maps, flags are cleared and a Toast is shown on others users end that "User _ has left the Call".

11 . If there are more then two users in the Audio Call, then even if one of them leaves the call(even being the initiator of the call), still others can continue their Call as only the leaving user flag is set false in their maps.





2.3.4 Interface Design:

2.4 CHALLENGES AND THEIR SOLUTIONS

- Audio call

  1 . Voice distortion

  2 . Lag in audio call

  3 . Network problem

  4 . On usage of thread there is lot of disturbance with 0 lag, on the other hand, if thread is removed there is lesser disturbance but lag is more.

5 . Not feasible options

* Using different ports for different users was not a feasible option for audio call
* Using different threads for different users due to disturbance

- Video Call

1 . Media recorder only record in mp4 and 3gp formats, but neither of these two formats provide streaming, i.e. they cant be recorded and sent through UDP packet at the same time as the header is missing from these type of files and is sent in the end so it cant be played at the other end.

2 . A different approach was tried to overcome this challenge. Small chunks of audio was recorded and chunks were being sent and played at the other end. (When whole chunk was received then only it can be played at the receivers end) and while playing it begins to receive the next chunk and appends it in the video view (Producer-Consumer Problem).

3 . Here the problem was that the Camera wasnt giving video as fast as it was being sent and played too.

4 . Thus to overcome the above problem, Frame was taken from the camera feed and audio is taken from the audio recorder and they are added to a buffer which is then sent as a UDP packet.

5 . The audio and frames are sent together to overcome the synchronization issue.

- Chat application:

1 . The 1st approached tried was sending messages through server, where the client will send the message and receivers name to the server and server will send that message to respective client at the other end. But then this approach was dropped and the chat between two users was changed to peer-to-peer chat to remove the role of server and reduce the server dependency of the overall application.

2 . A group chat feature was also added. But here, the problem was to make every other user of the group chat know that they have been added to a chat room. So they are sent IP addresses of all the people in this group. To prevent the privacy of this group, a group code is made using the initials of all the users of that particular group chat and it is sent to other members. So this group code acts like a primary key for

a particular group. And thus no other user outside this group can join this group as that user wont be having this groups group code.

- Server

  1 . create a database which stores the username and password information of all registered clients.

  2 . connect to the database using jdbc driver.

  3 . Create a UDP socket to receive messages from the clients.

  4 . Create a new thread. In this thread- (Server Time)

  * create another UDP socket to receive timestamp requests from the clients.
  * receive the time message.
  * send back the current time of the server.
  * go to second step

  5 . Create another thread. In this thread (OnlineChecker)

  * sleep for 2 sec.
  * get the IP addresses of all online users from the database.
  * For all the IP addresses obtained:
    Ping each IP address.
    If reply is received, do nothing.
    Else, update the database to mark the user as offline.
    Go to first.

  6 . In the main thread:

  * Receive the incoming message, MESSAGE
  * If (timestamp of the MESSAGE) is BEFORE (servers current time) go to 5.1
  * If MESSAGE=login + username, update the database to mark the user as online.
  * Else if MESSAGE=logout + username, update the database to mark the user as offline.
  * Else if MESSAGE=chat—file—audio—video— + username + remote username , if the user with received username is online, send back the current IP address of the requested user. Update the availability status or busy field in database

- Else if Message = group + username +remote user(i), if the user with received username is online, send back the current IP address of the requested user.
- Go to first.

2.5 SUMMARY AND CONCLUSION

2.5.1 Summary

We started to build our Application with Group Chat via a Server. Then we moved on to share files across Aakash Tablets Peer-to-Peer. Our Server was then enhanced for user registration and authentication. As we already were in possession of "Video-Chat Software", we started working on improving the audio quality and minimizing the lag. Different configurations for AudioTrack and AudioRecord were explored till we arrived at a distortion free audio call. We tried different approaches for Peer-to-Peer Video calling but we got success by sending the camera feed in frames embedded with audio through UDP sockets to achieve synchronisation. Then we worked on Peer-to-Peer Group Chat and Audio Conferencing. Along with these we started working on improving the UI of our Video Conferencing application. We integrated all the different features being developed together in one single application. We also included feature of setting the IP of server. The built application is approached to achieve real time environment.

2.5.2 Further Enhancements

1 . Video Conferencing.

2 . Improve Video Quality and reduce the size complexity of frames being sent.

3 . Adding Security by encrypting messages, files, camera feeds and audio feeds and adding session.

4 . Incorporating Database for storing messages of every Group Chat Session.

5 . Enhance features of the server like handling concurrent requests,session maintenance etc.

CHAPTER NO. 3

USER MANUAL

3.1 CLIENT

3.1.1 INTRODUCTION

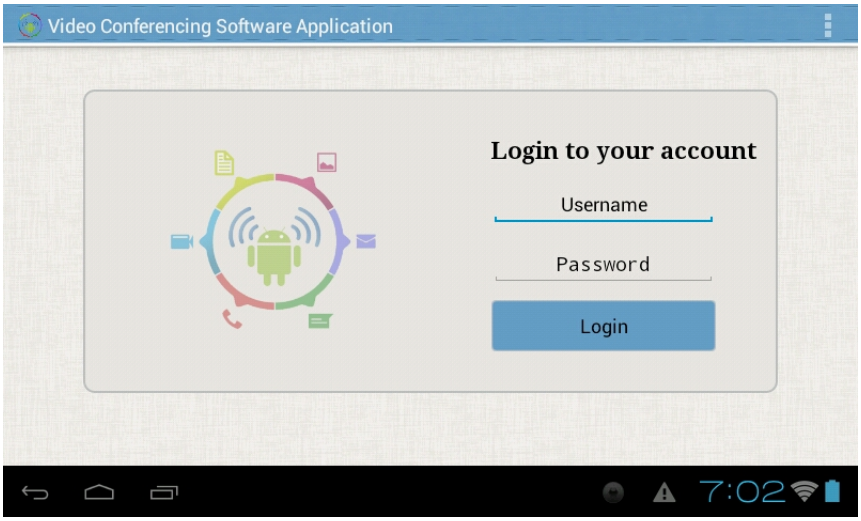
Through “Video Chat Application” file sharing, audio calls, video calls can be done between two tablets running on Android OS 4.0.3 or above. Multi-User Texting and Audio conference is also possible.

3.1.2 STEP 1: Start the application

When you switch ON the tablet you will see on the screen, the display similar as shown in the Figure below. Click on the launcher icon provided on the right upper corner of the screen.

Then you will see on the screen all the application present in the tablet.

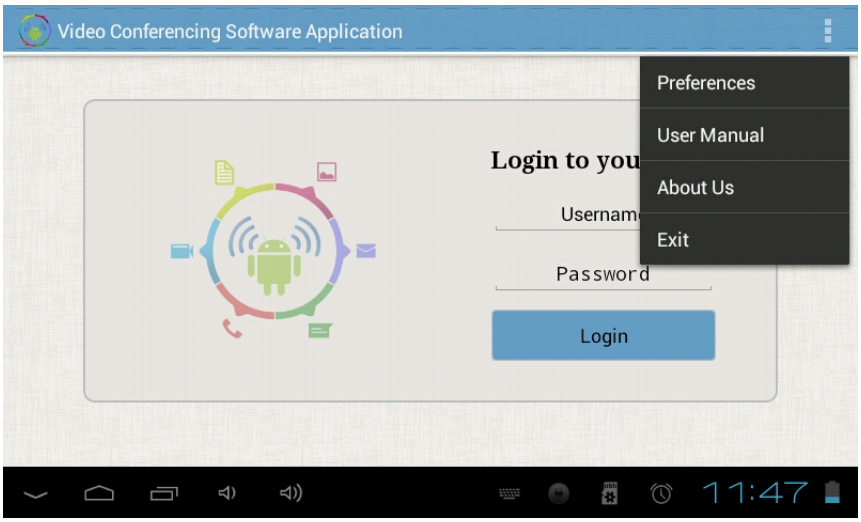
Click on "Video Chat" icon present in the tablet as shown in the figure below. Login page of the application appears



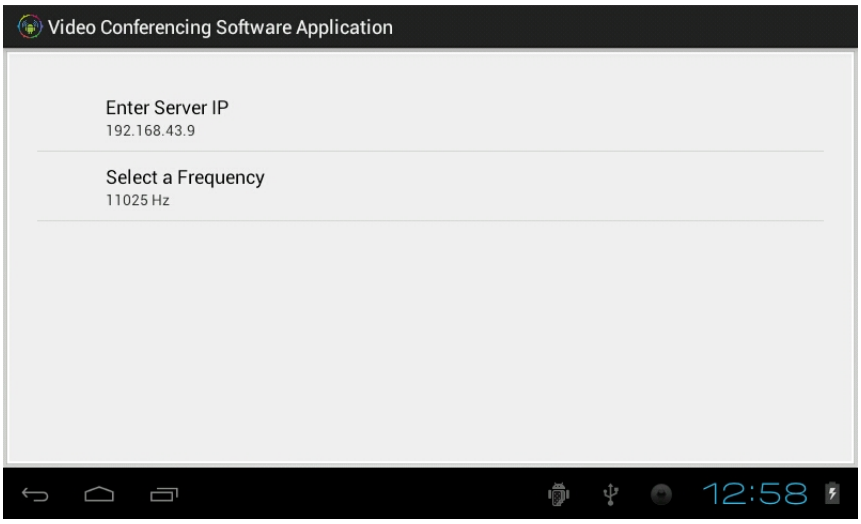
3.1.3 STEP 2: Setting up the application

- *Set up the server IP Address*

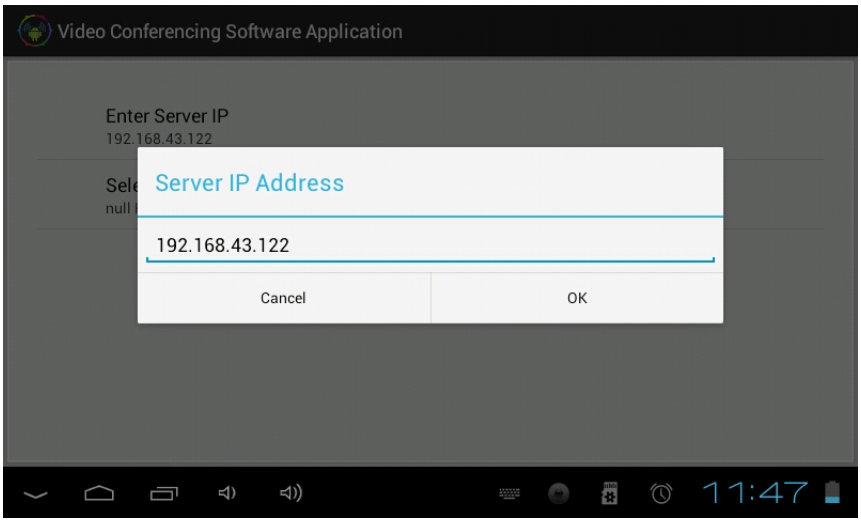
Click on the options menu on the right upper corner



Select the preferences. You will see the following dialog as in Figure. It will have a default IP Address, if the application is run for the first time or the previous IP Address which was set.



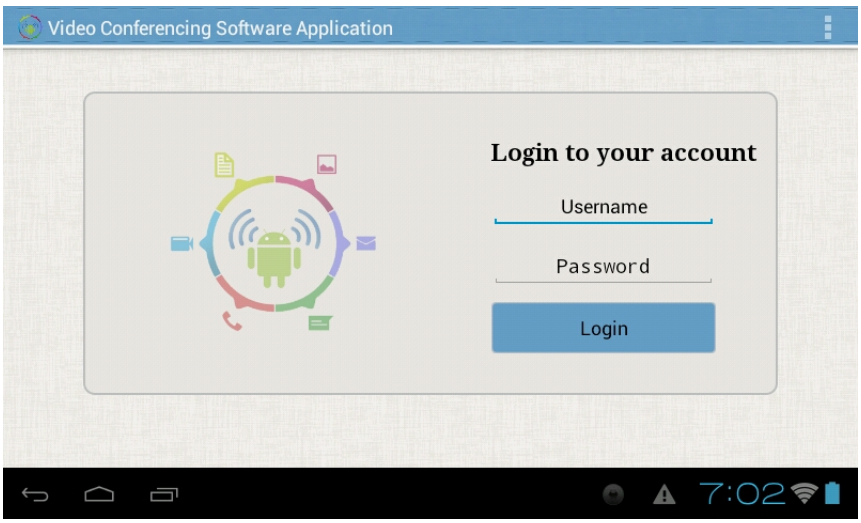
If you have to change the Server IP Address click on the Server IP Address. Next you will see the text box to enter the Server IP Address as in following figure



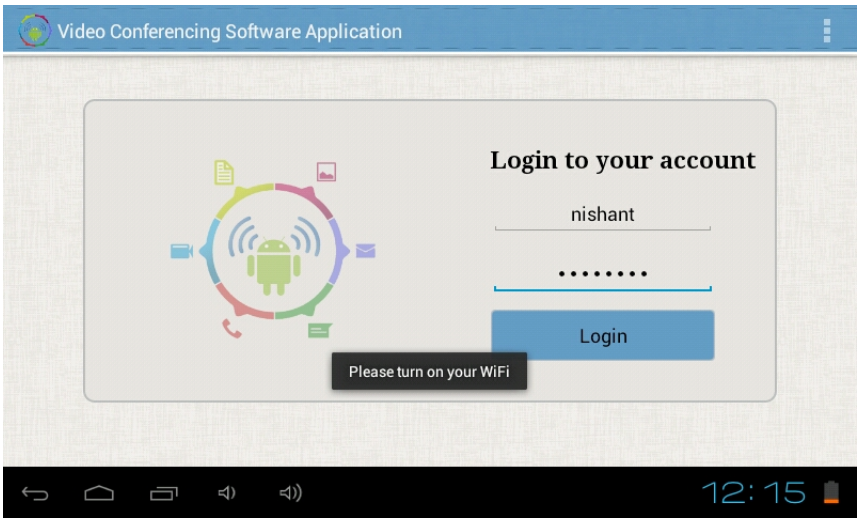
Enter the desired Server IP Address as provided by the system administrator and click OK.
NOTE Server is used for user authentication.

3.1.4 STEP 3: Logging in to the server

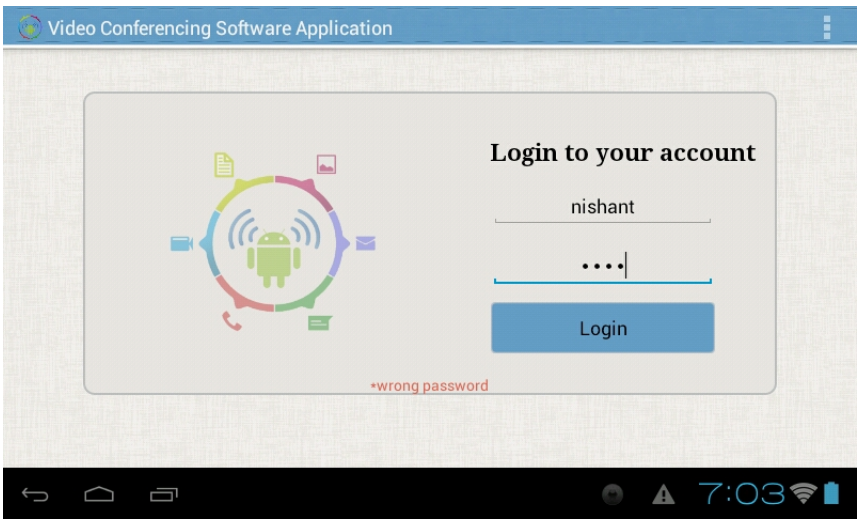
After setting proper parameters , Type in your username and password as provided by sys-admin[Refer. User Manual Client Registration on server] Then click on the Login button as in Figure



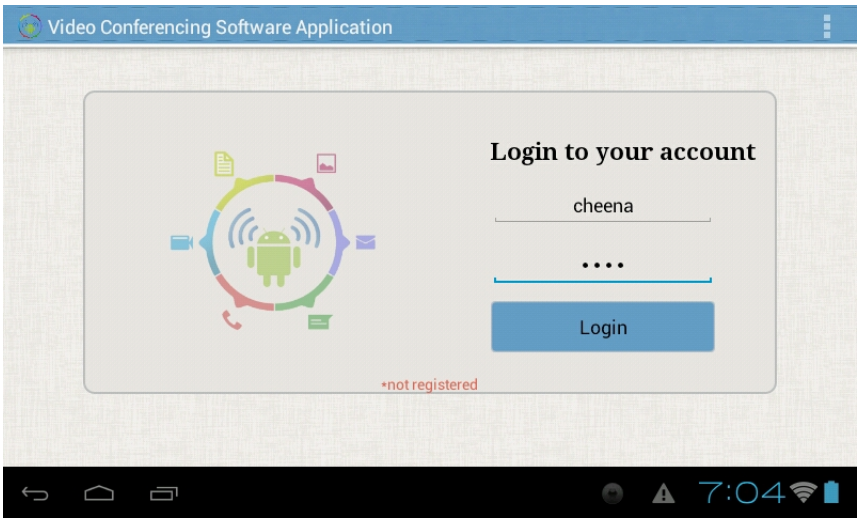
- If wifi on your device is not running you will get a message to turn on your WiFi Like the figure below



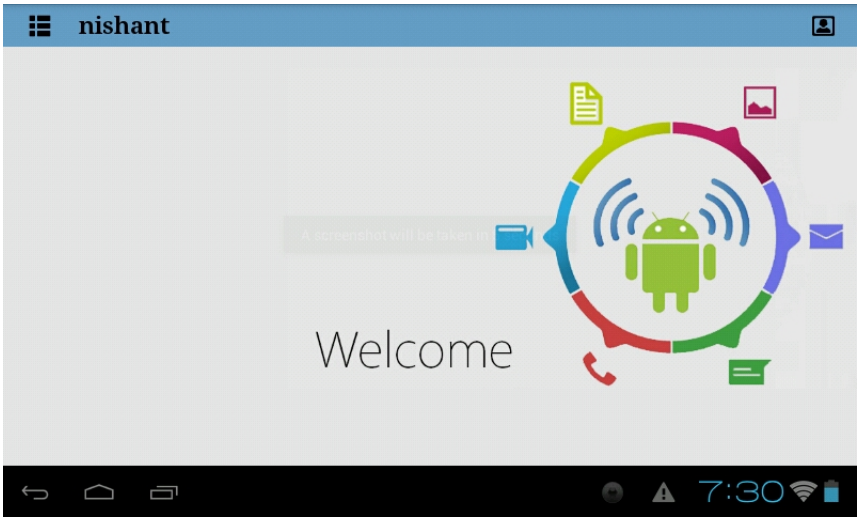
- If you have provided wrong password you will get message : wrong password as the figure below:



- If you have provided unregistered username you will get message : username is not registered as the figure below:

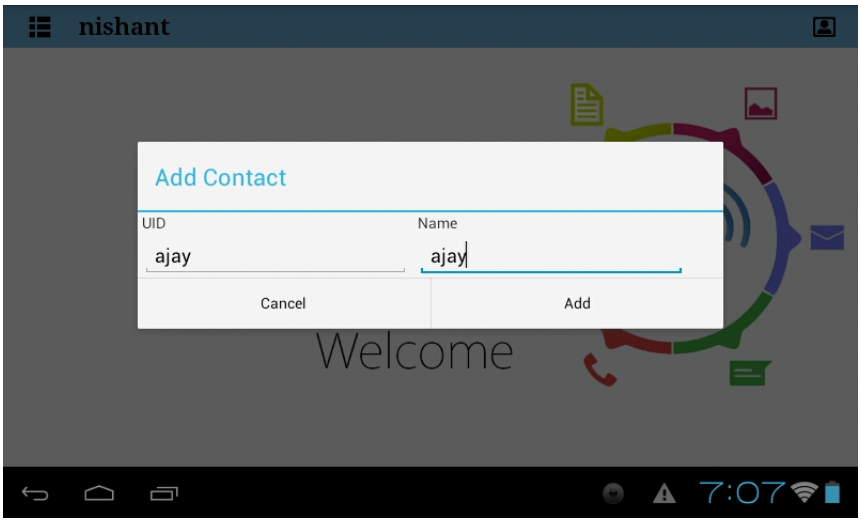


- If server is running on your network and your wifi is on you will get connected with it Given the fact that you have provided correct username and password. The home screen opens up as shown below:



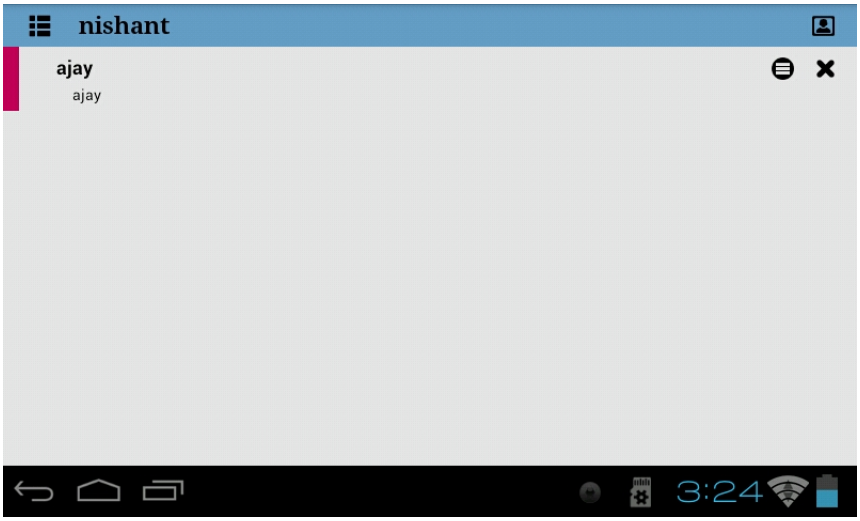
3.1.5 STEP 4: Adding Contacts

On home screen an icon is present on the right upper corner. This icon is used to add contacts. On clicking the above button you will get a dialog as shown below



Enter the unique id and name of the contact already registered on the server

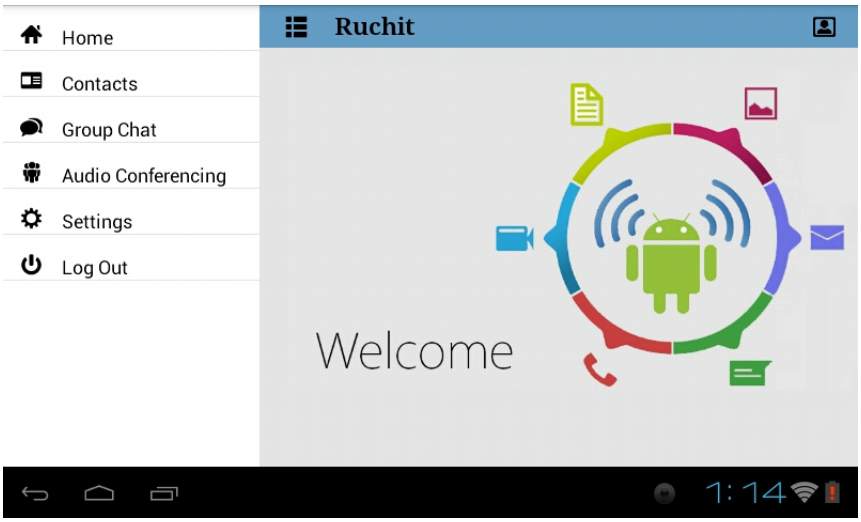
- If you do not enter either of the fields you will get a message: enter proper details
- If both fields are filled then on pressing the add button, the particular contact is added to your contact list.



- On pressing cancel home screen comes back

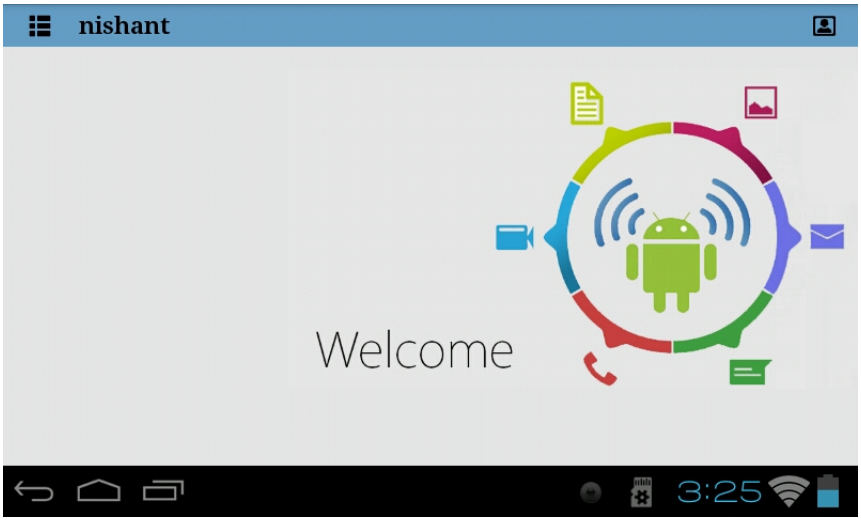
3.1.6 STEP 5: Select an action

On home screen an icon is present on left upper corner. On pressing this icon the menu appears from the left side with 6 menu items



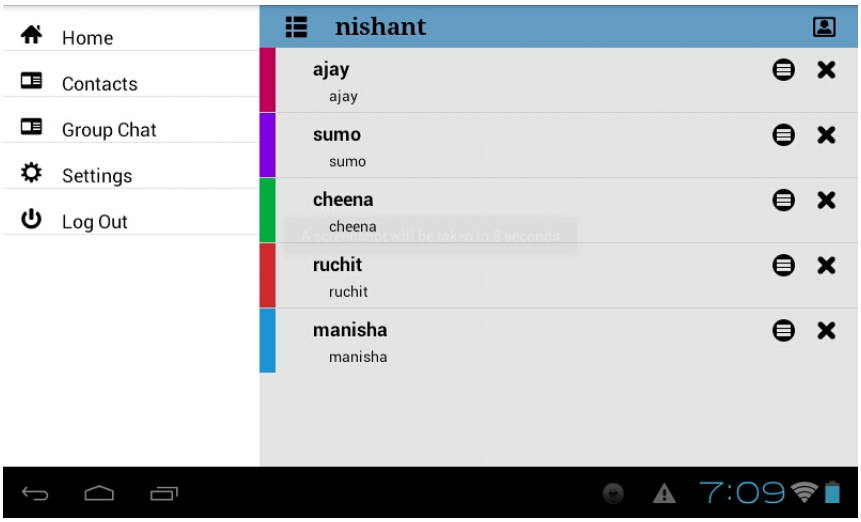
- *Home*

On Pressing Home, user comes to home screen of the application.

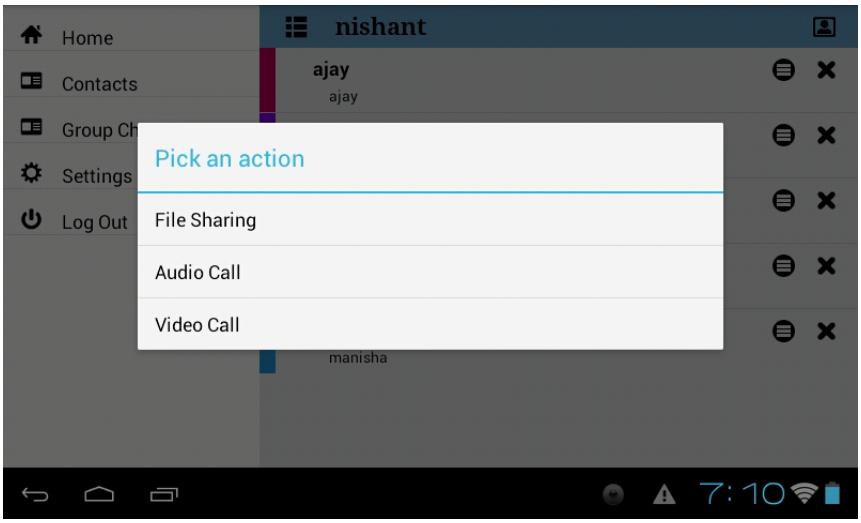


- *Contacts*

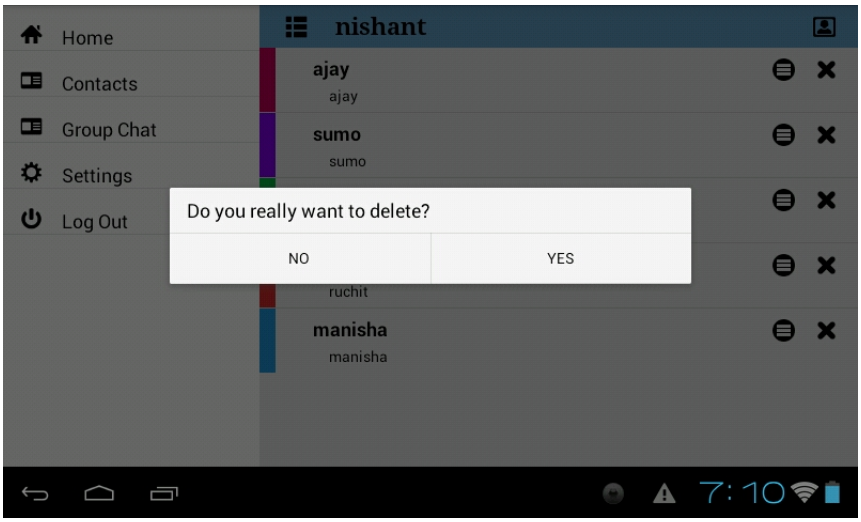
As the Contacts menu is selected, the user can see all the contacts saved on his device



On the left side of each contact there are two buttons. First button is for action menu. Clicking on the first button, an action menu opens up as shown below, user can select any of these options.

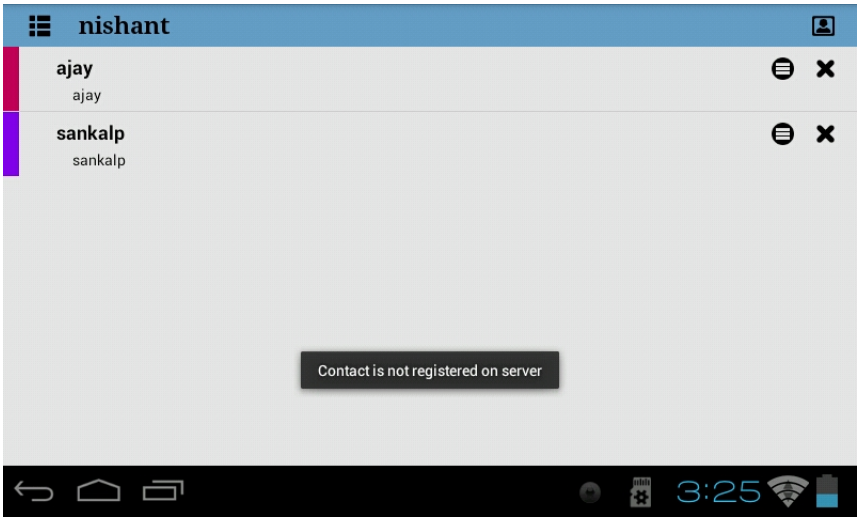


second button on contact is to delete that contact, If user clicks on this button a confirmation box opens up as shown below. If user selects yes, contact is deleted from his contact list.

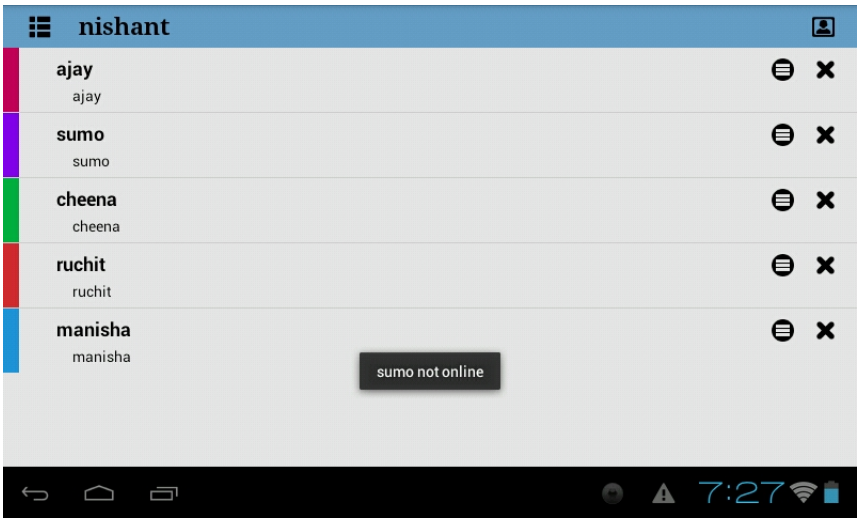


User can select audio call, video call or file transfer

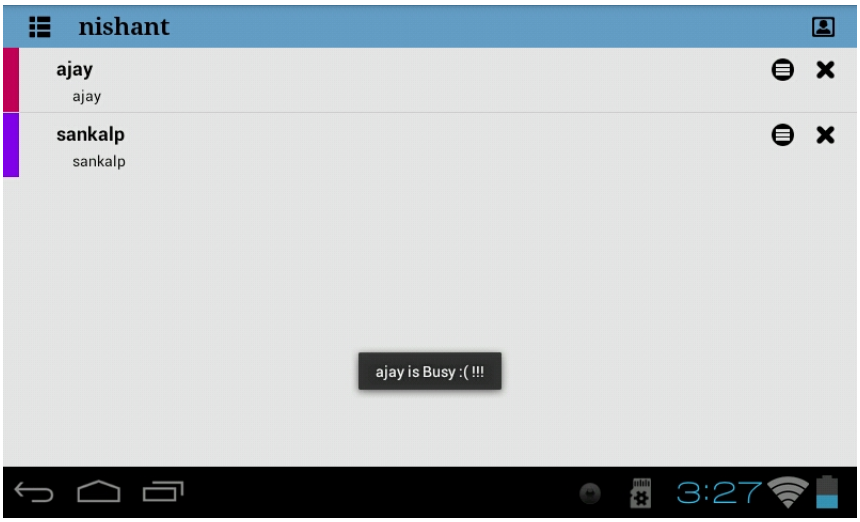
- If selected user is not registered on server. User get the message : User is not registered



- if remote user is offline User get the message : User is not online.

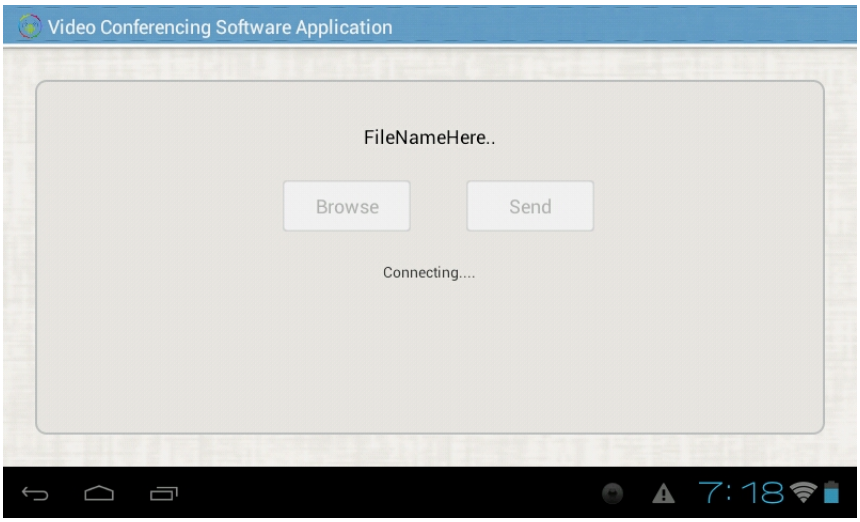


- if remote user is busy. User get the message : User is busy.

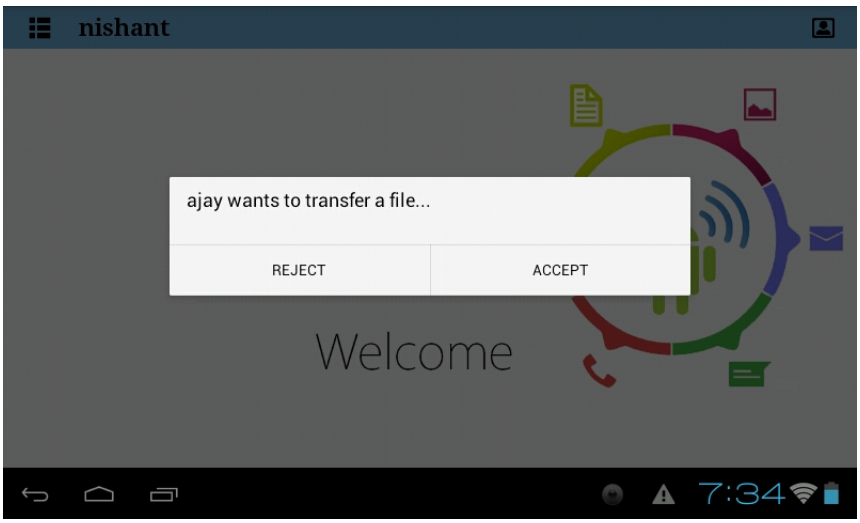


- if user is online then

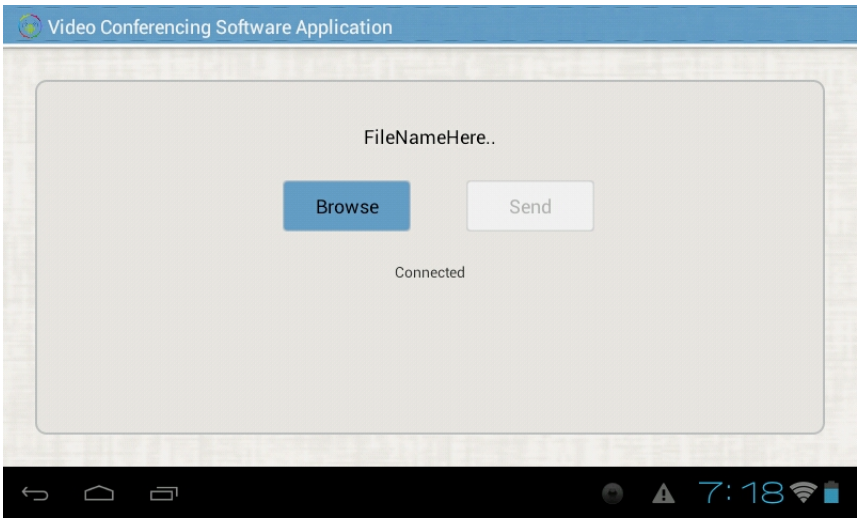
File Transfer if user selects the file transfer following screen opens up



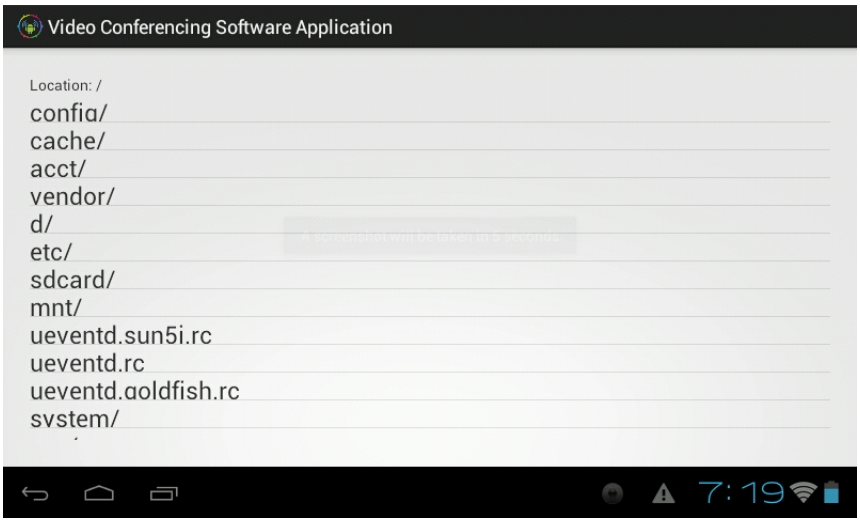
remote user gets a file transfer request.



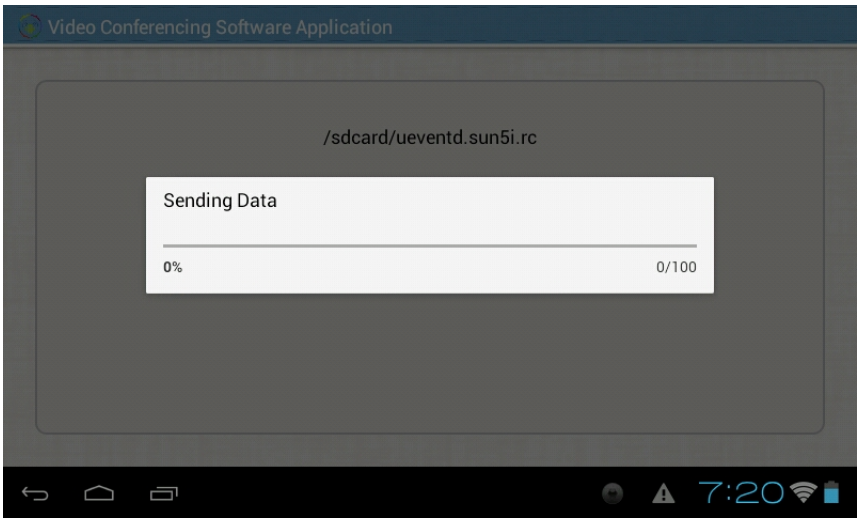
If remote user accepts the request following screen appears.



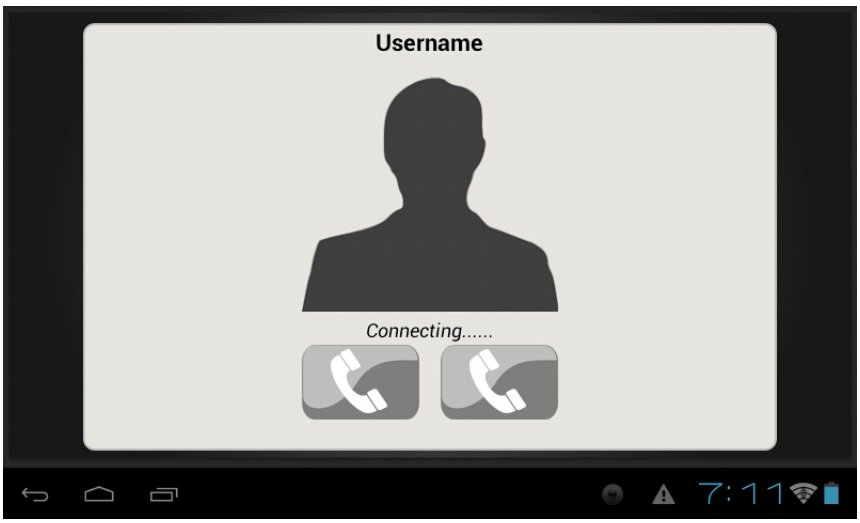
It has two buttons ‘Browse’ and ‘Send File’. Now user can browse a file from internal or external memory to transfer by clicking the browse button



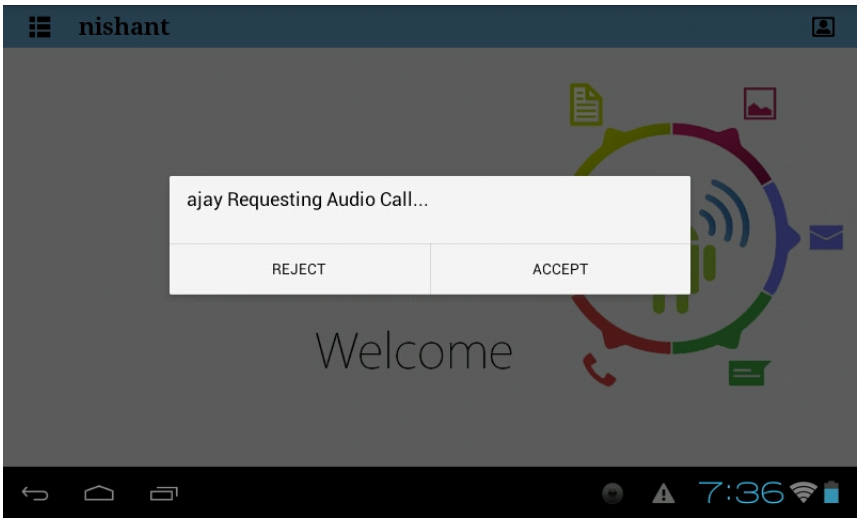
After selecting the file, user click on ‘Send file’ to transfer. Now a box appears, Which shows the amount of file transferred.



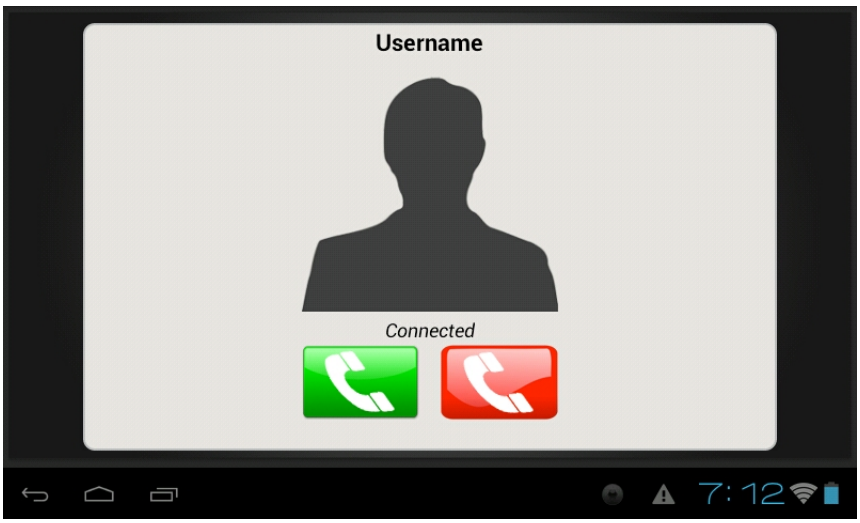
Audio Call If user selects the audio call following screen opens up



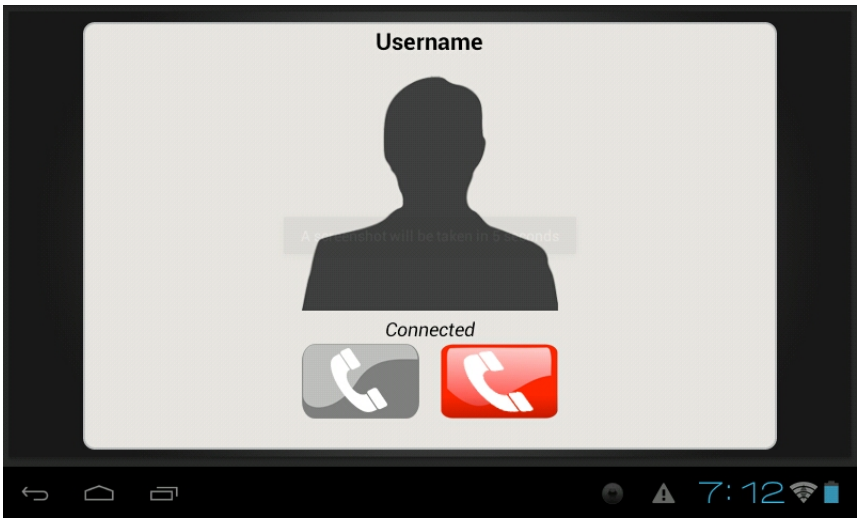
remote user gets a audio call request



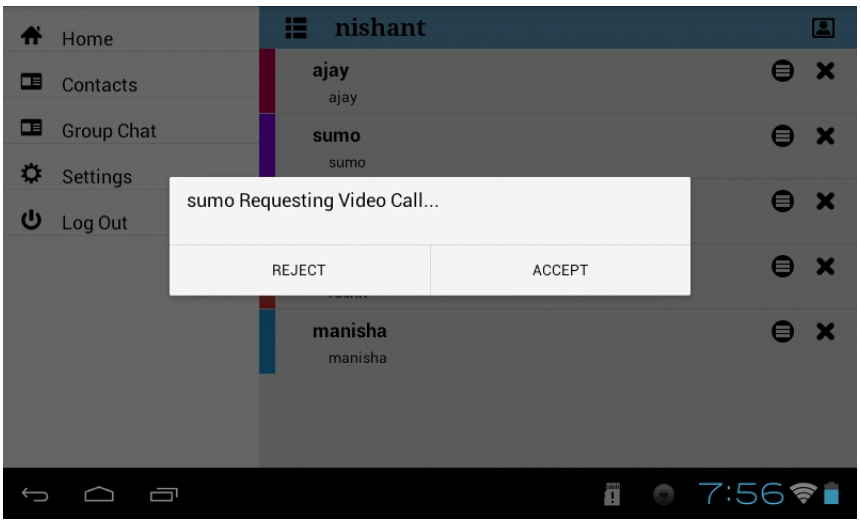
If remote user accepts the request following screen appears. Status is changed to 'Connected'



It shows two buttons to start and stop the audio call. Now user has to press start button to start audio call. To stop the audio call the stop button has to be pressed. If user rejects the call status text is changed to 'Call has been rejected'



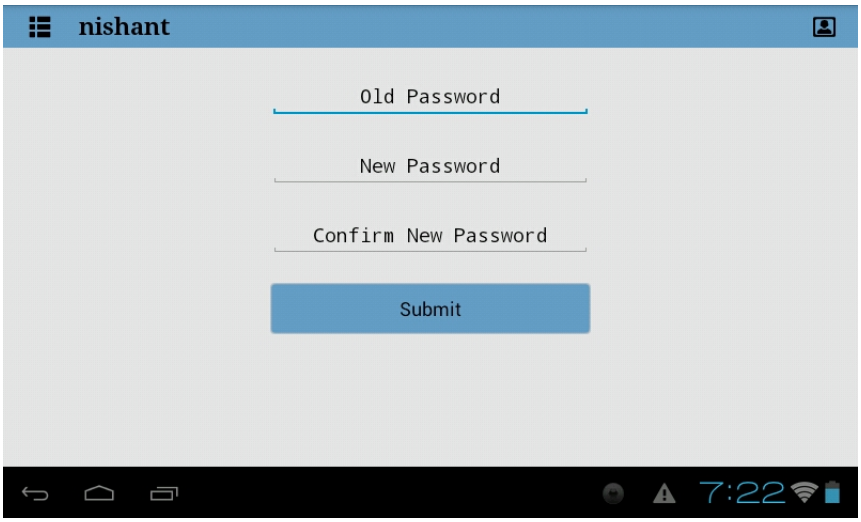
Video Call If user selects the Video Call and remote user gets a video call request



if remote user accepts the call, call status is changed to connected from connecting. It shows three buttons to Start, Receive and Stop the Video call. Now user has to press start button to start the camera. And receive button to receive the video stream of the remote user. To stop the video call the stop button has to be pressed.

3.1.7 STEP 6: Change the password

To change the password user has to click on Settings in the menu. Following screen appears

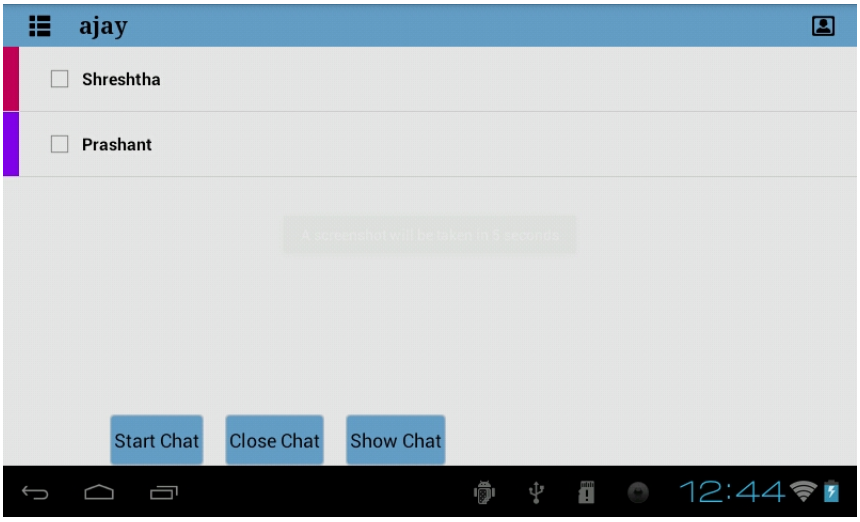


User has to enter his old password, new password and confirm password, then user has to press submit button.

- If user entered wrong old password, It displays a message : Wrong old password
- If new password and confirm password doesn't match, It displays a message : Password mismatch
- If user enters correct old password and new password, confirm password matches then password is changed successfully.

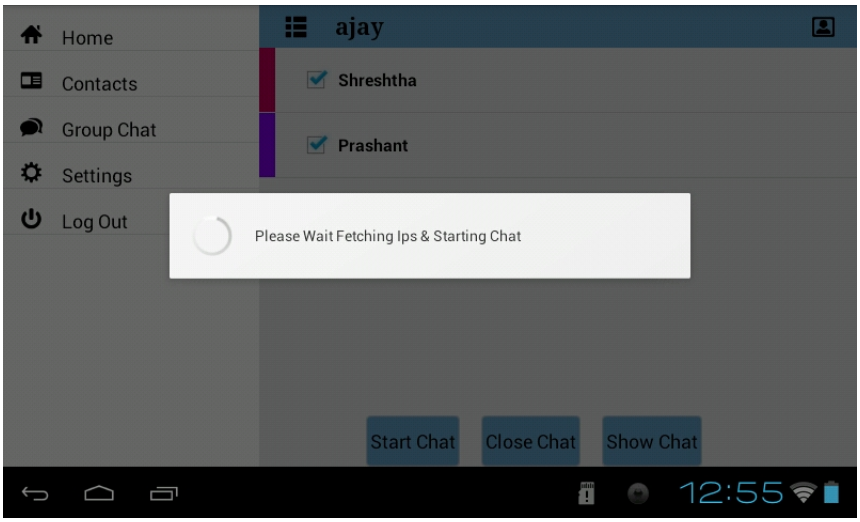
3.1.8 STEP 7: Group Chat

- Group chat Menu showing the list of contact added options for Start, Close and Show chat.

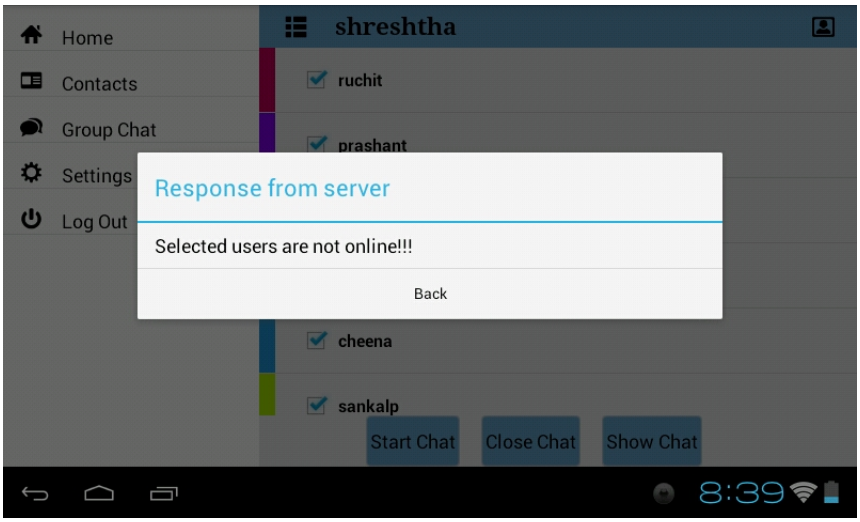


- Select a any number of contacts from the contact list and start the chat.

- Requesting server for IP address of users and starting the chat.

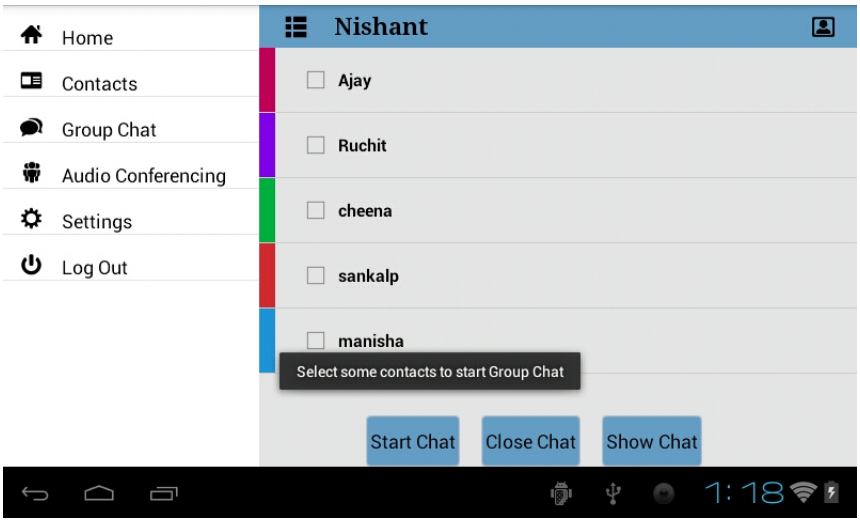


- If no selected contact online in the list

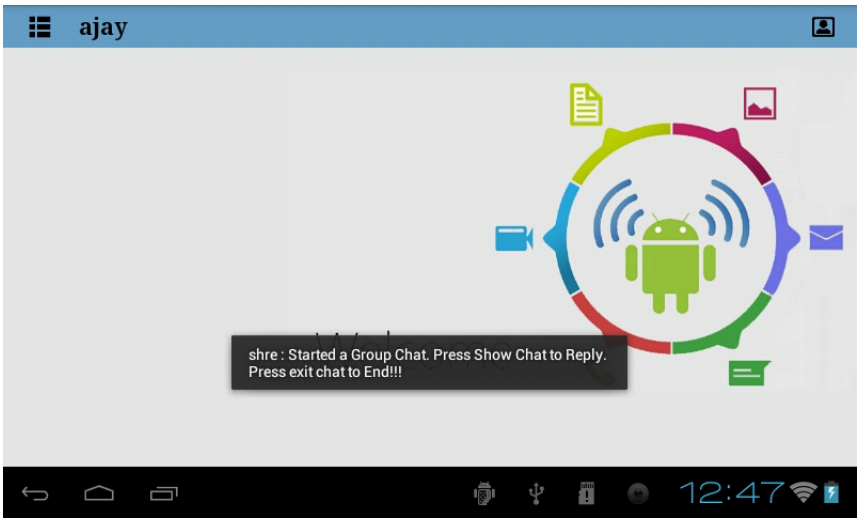


- If they are available to chat, a chat room is started informing all the users about the chat.

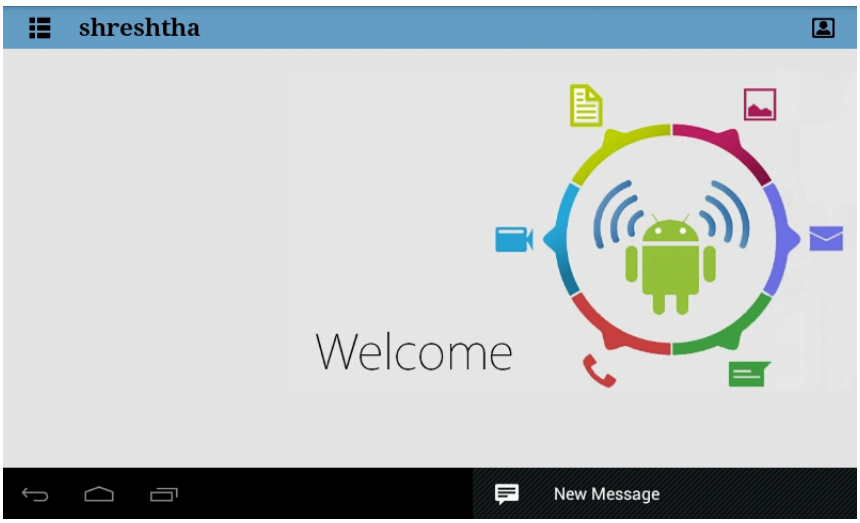
- If no users are selected and chat is started a notification is shown.



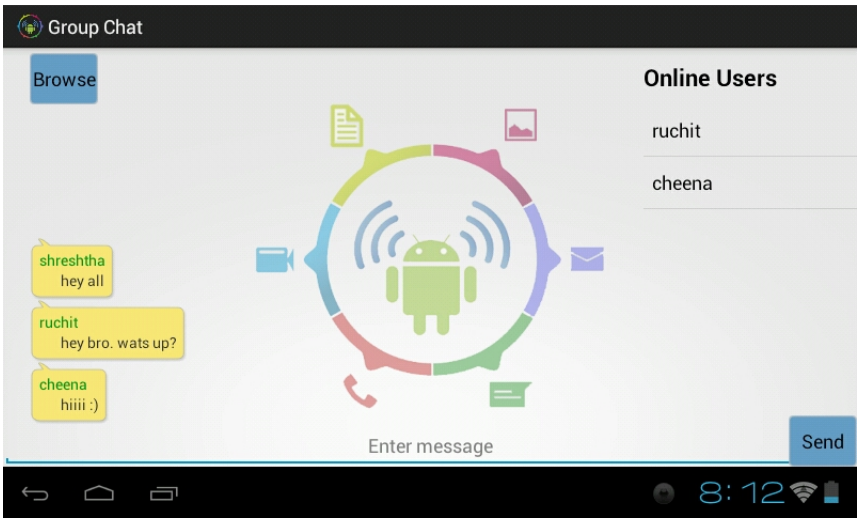
- A groupchat staring notification.



- Notification on homescreen for a new message.

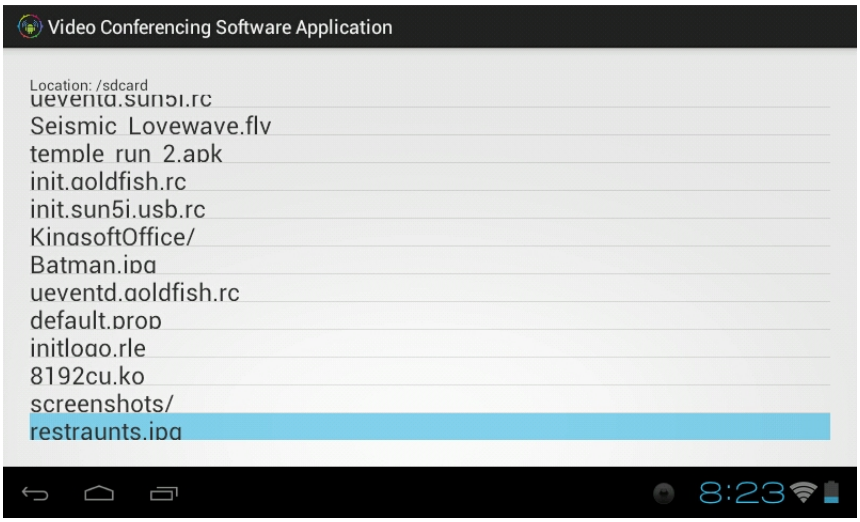


- Now the users in the room can chat through typing in textbox, the messages are sent to all the user in the room.



- New messages in the room appearing.

- Selecting a file by clicking the browse button

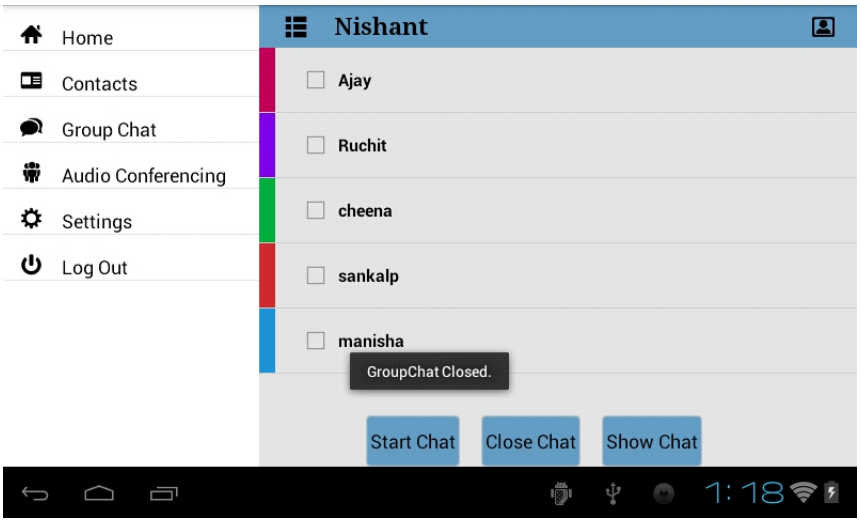


- Sharing a file in the room, makes it availavle to everyone online.

- A snapshot of received file from the conversation.

- A part from coversation.

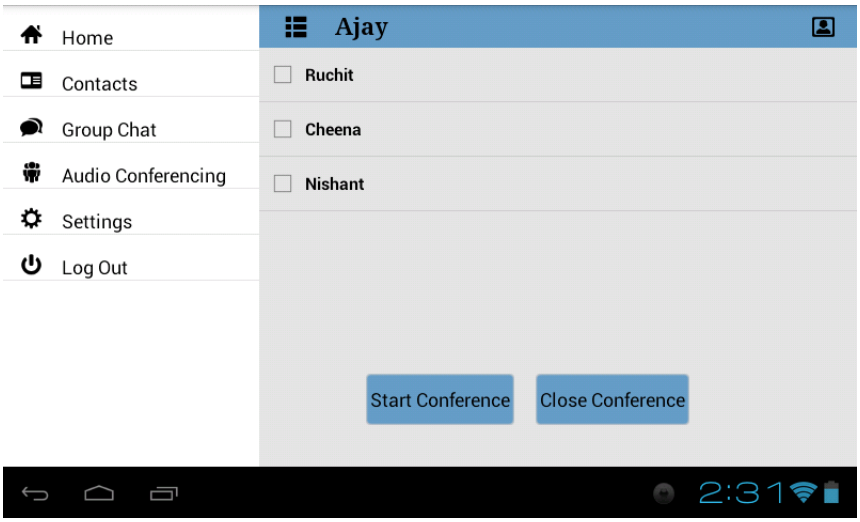
- Closing a group chat



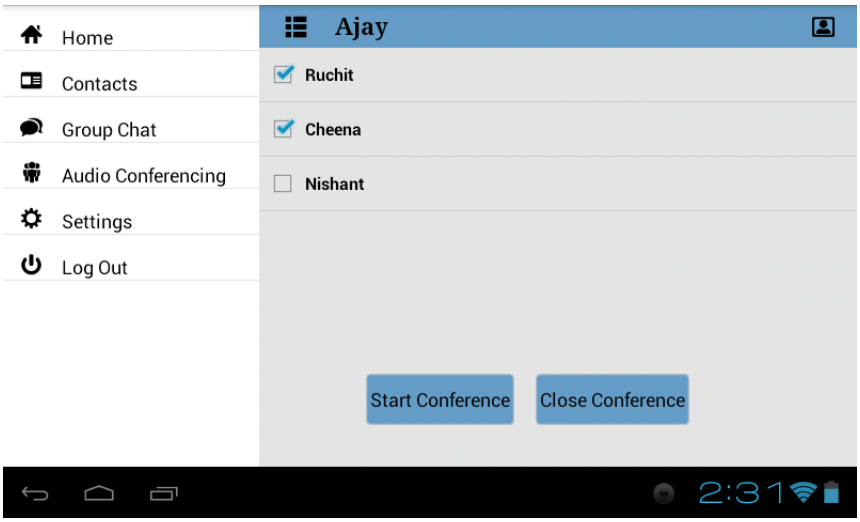
- User leaving the room appearing messages

3.1.9 STEP 8: Audio Conferencing

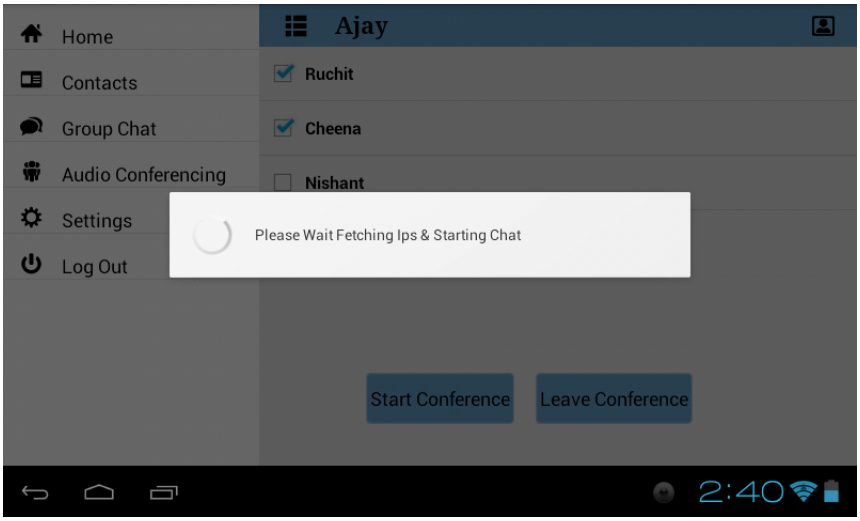
- Click on the audio conferencing option in the menu bar of the application. A screen opens with the names of your existing contacts , with a checkbox besides each contact.



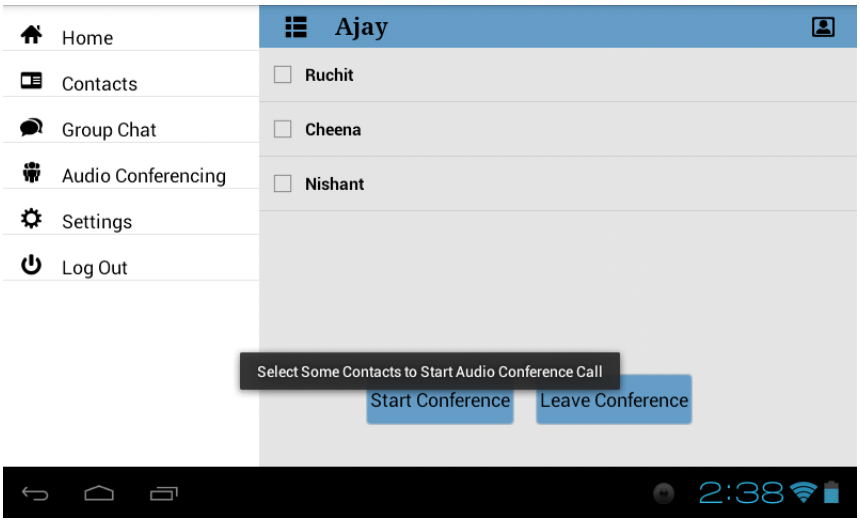
- Tick the checkbox of the users(in your contact list) with whom you want to start the audio conference.



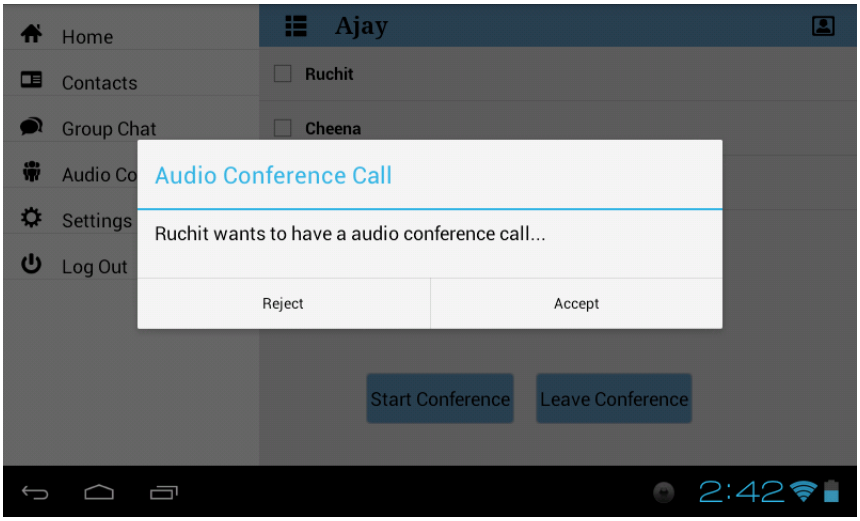
- Press the start conference button , to send a conference request to all selected users. The following progress bar will appear on your screen



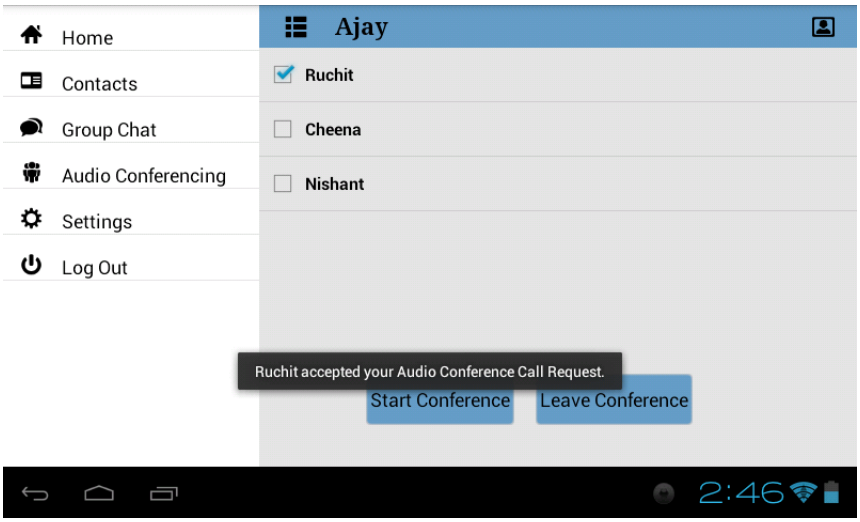
- indicating that you are receiving IP's of selected users from the server. If you press the start button, without selecting any contacts, the following toast will be shown on your screen.



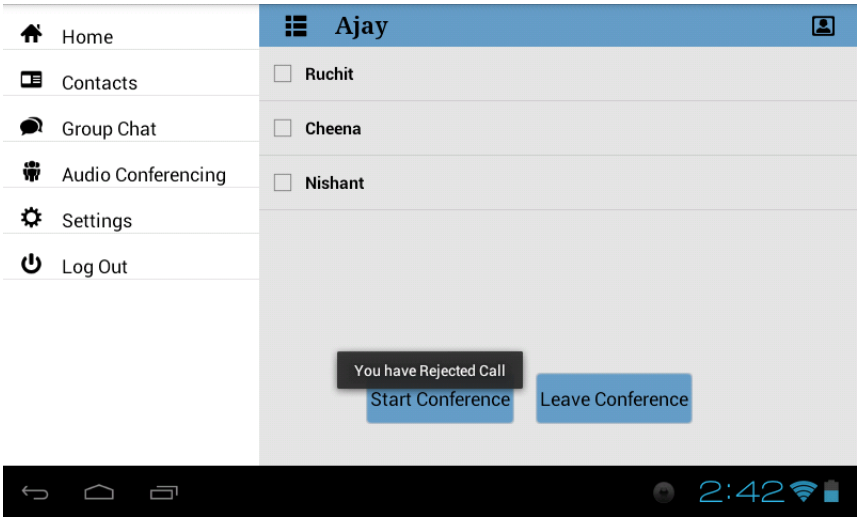
- When you send a conference request to a user, the following dialog will appear on the user's screen.



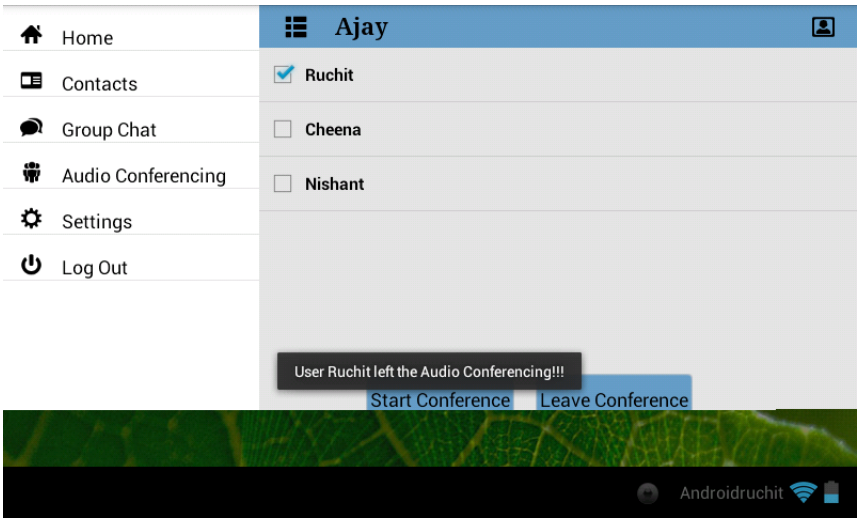
- If the user presses the "Accept" button, you will get the following toast, indicating that the other user has accepted your call.



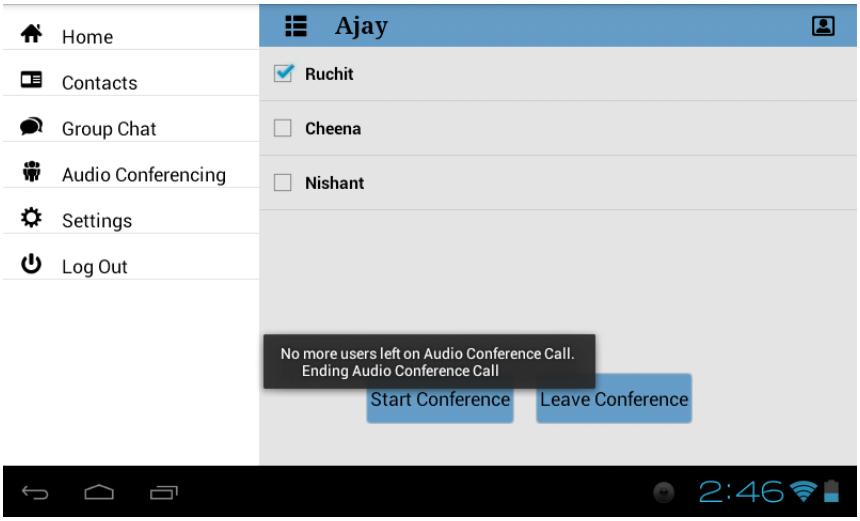
If the user presses the “Reject” button, you will get the following toast, indicating that the other user has rejected your call.



- If any user leaves the conference in between(while you are still in the conference) , then the following toast will be displayed indicating that , that particular user has left the conference call.

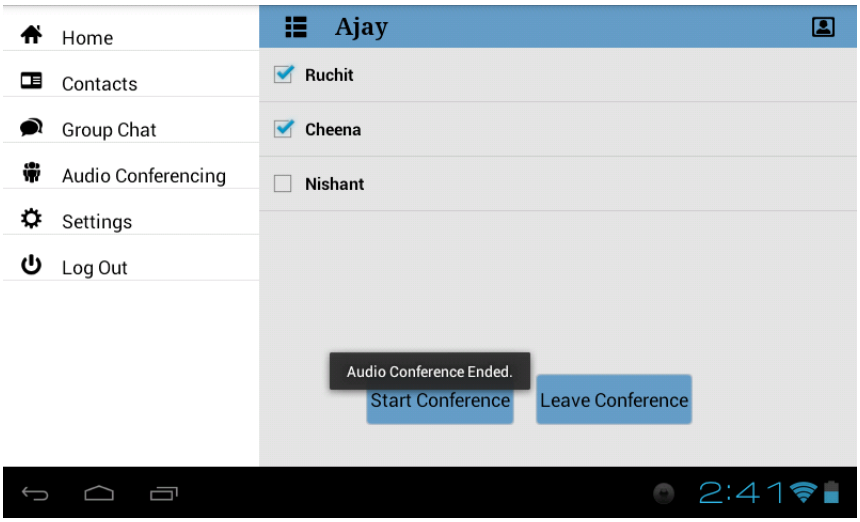


- If all the users leave the conference(while you are still in the conference,i.e you have not ended), then the following message will be displayed on your screen



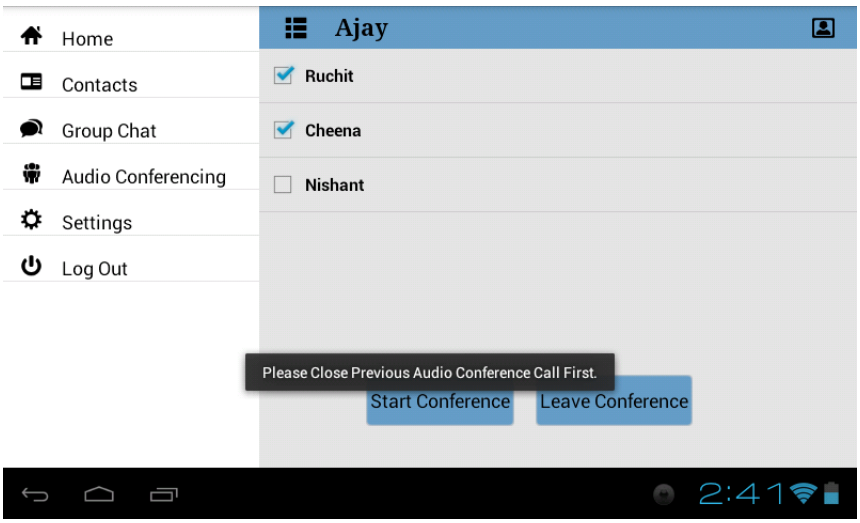
indicating that all users have left the conference, which means the conference call has ended.

- If you want to leave the conference at any point of time, you can press on the “leave conference” button, following which a message will be shown on your screen,



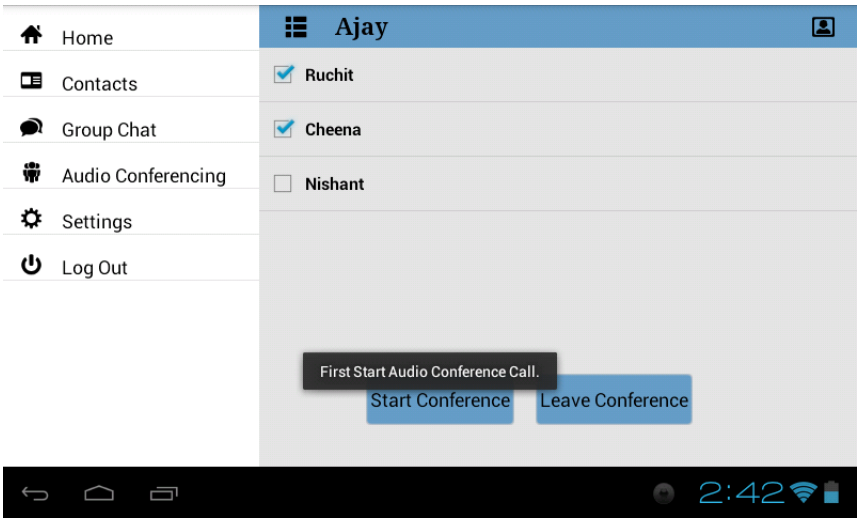
indicating that the audio conference(for you) has ended. If you leave the conference , all other users ,which were in the conference, still continue to communicate.

- If you press the “start conference” button while you are still into a conference, the follwing message will appear,



indicating you need to first close one conference to start another conference.

If you press the “leave conference” button , even though you are not in any conference, then you will be given a message, as shown in figure

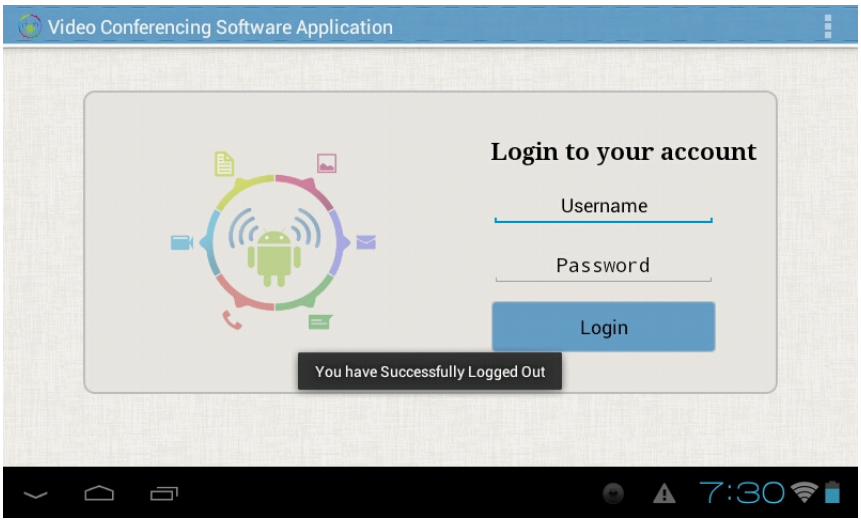


indicating you must first start a conference, to leave it.

- After you have left a conference call, you can again start a new conference call now, by following the procedure ,to start a conference, mentioned in this document.

3.1.10 STEP 9 : Logout

To logout from the server, user has to press logout in menu item. After logging out from the server, login screen appears with message: you have successfully logged out



3.2 DESKTOP SERVER

3.2.1 INTRODUCTION

The server is used primarily to maintain a database which has data regarding the clients and which can be manipulated according to the requests of clients. The server registers the clients with unique username and password. To get the server running on a machine, it must have mysql 5.5 server installed in it as well as the JDK 1.6 or above

3.2.2 STEP 1: Run the Jar file

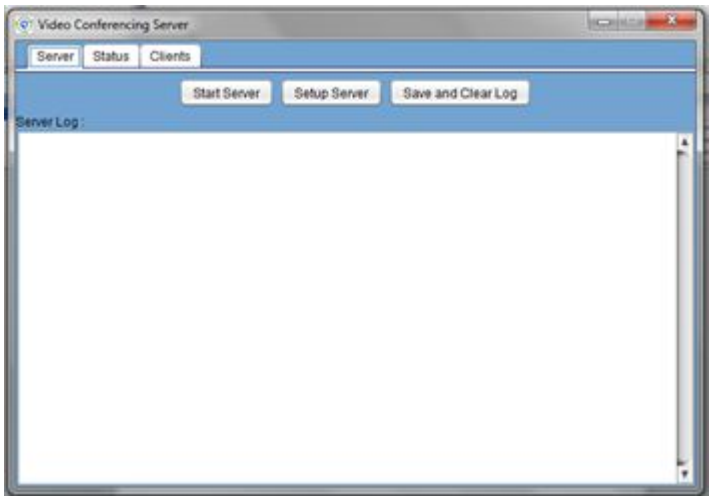
On starting the java application for the first time, it asks the administrator to set up an Admin password

3.2.3 STEP 2: Set up Admin password

- Case 1: If there is a password Mismatch then a dialog box shows up as shown

- Case 2: If Exit using Close button on the right upper corner the application stops and on restarting again asks to set the Admin password

- Case 3: If Admin password is correctly set up then the Server Main Frame opens



NOTE: At any point of time if you close the application and restart the application after setting up the password, the Admin Login Window appears

- Refer to Subsection 3.2.7

THE MAIN SERVER FRAME

The server frame consists of 3 panels

Server Panel

The server panel has a server log and three buttons namely

1. Start Server
2. Setup server
3. Save and Clear log

It also contains a server log which keeps track of the operations and requests honoured and denied by the server

- On clicking the start server button
  Two things can happen:

  Case 1 : if you haven't setup a server then on clicking start server, the log shows messages as follows:

"ERROR: Table 'userdetails.address' doesn't exist"

Case 2: if u don't have the required mysql driver then error log comes as:
"ERROR missing 'com.mysql.jdbc' driver"
To get rid of these errors the admin must set up the server first to do so click on Setup Server button

3.2.4 STEP 3: Setting up the mysql server connection

On clicking the setup server button following screen opens up

The admin has to fill the informations about the server correctly, so as to connect with the mysql server through JDBC otherwise the previous errors would persist after proiding the correct details press the save button

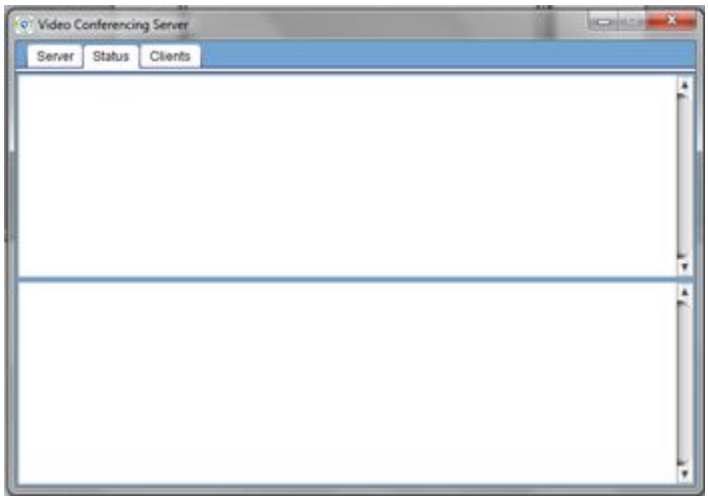
- and then click the create database button and the create Table button
- now click on the close button to return to the server main frame
- observer that the previously coming errors on the server log has stopped
- clear the log by pressing the Save and Clear log button to refresh the log

NOTE: At any point of time if you close the application and restart the application after setting up the password, the Admin Login Window appears

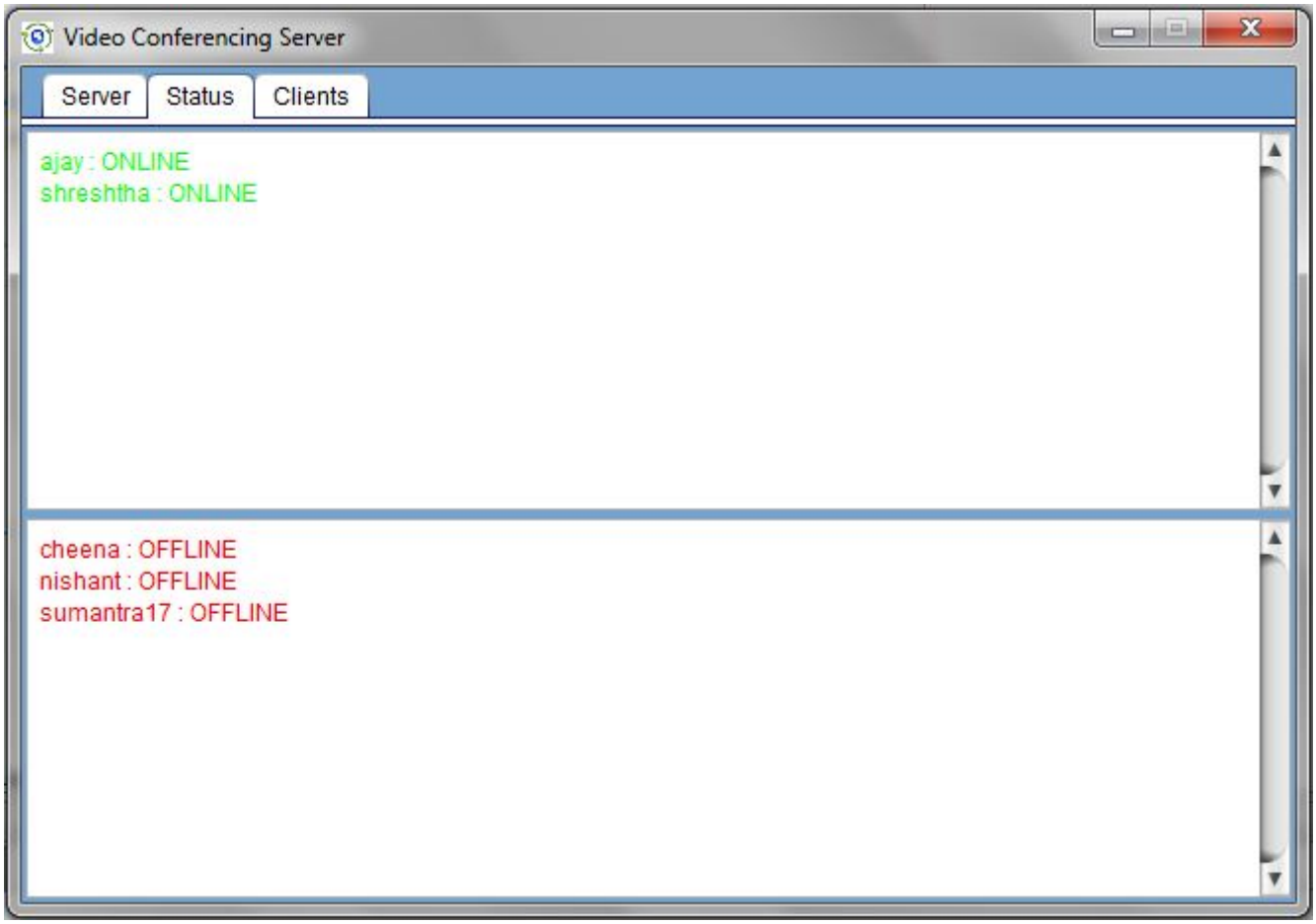
- Refer to Subsection 3.2.7

Status Panel

Unless any user is registered by the Admin the panel looks like



The upper part shows the registered and online users and the lower part shows the registered but offline users



To see this functionality let us register a user

3.2.5 STEP 4: User Registration

The Registration panel holds two fields in which username and password could be set for a client
it also has 4 buttons

1. Save
2. Delete
3. Kick
4. View Database
   On clicking the save button following things can happen

Case 1: If no username is given a dialog box comes as shown below

Case 2: If no password was selected a dialog box comes as shown below

Case 3: If a username already exists a dialog box comes as shown below

Case 4: If unique username and a password is given

a dialog box comes as shown below

On clicking the delete button

- Case 1: If inappropriate username is given
  a dialog box comes as shown below

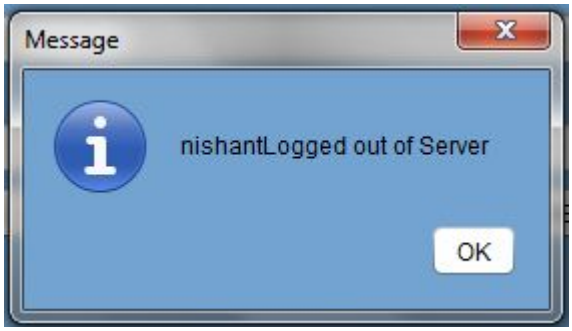
Password is not required to be given
On clicking the kick

- Case 1: If no username was given
  a dialog box comes as shown below

- Case 1: If inappropriate username is given a dialog box comes as shown below

- Case 2: If unregistered name was given a dialog box comes as shown below

- Case 3: If registered name was given a dialog box comes as shown below



3.2.6 STEP 5: View The Databse

On clicking the View Database Button the present state of the table can be viewed by the admin

3.2.7 ADMIN LOGIN AND PASSWORD CHANGE

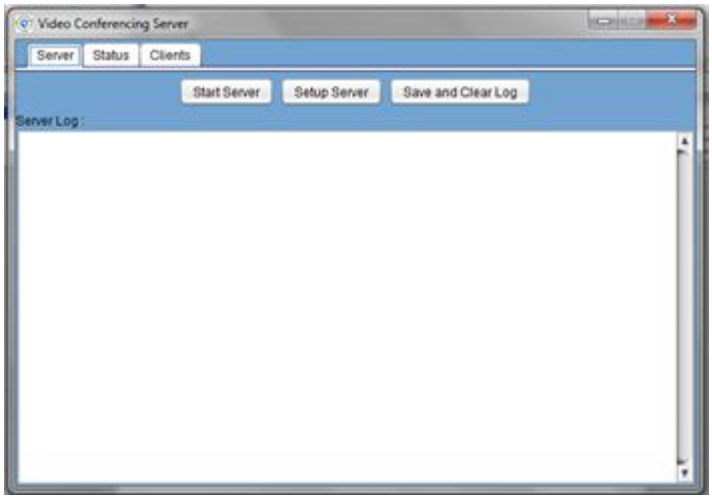
On starting the java application for the second time given the fact that the admin password has been set up, it asks the administrator to login as Admin by giving the Admin password

- Submit Button Functionality

  Case 1: If the password is wrong then a dialog box shows up as shown

Case 2: If Exit using Close button on the right upper corner the application stops and on restarting again asks for Admin password

Case 3: If Admin password is correctly given then the Server Main Frame opens



- Change Password Button Functionality:
  If the change password button is clicked, following frame opens up

The user has to enter the old password, then the new password and at last confirm the old password and press on submit On clicking following things can happen:

Case 1: If old password is wrong then message comes

Case 2: If confirmed password didn't match the new password then message appears

Case 3: All fields are correct and the message appears:

And opens the admin login again

On submitting the new password the server frame opens up

Follow Step 2;

References

[1] Tutorial Videos:
thenewboston.org
youtube.com

[2] Resource Book:
Mark L. Murphy panion
Begining Android

[3] www.developers.android.com www.stackoverflow.com/questions/tagged/android www.stackoverflow.com/questions/tagged/sockets www.code.google.com www.github.com www.docs.oracle.com/javase/tutorials/networking/sockets/definition.html www.docs.oracle.com/javase/tutorials/essentials/io/index.html

[4] IDE Used:
Eclipse
Netbeans

[5]