TALKFREE

Mobile VoIP Application

Prepared by

NC Summaya Mumtaz

NC Shajiha Javed

Capt Yasir Hussain

Supervisor:

Dr Asif Masud

Submitted to the Faculty of Computer Science

National University of Sciences and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering

June 2012

CERTIFICATE

Certified that the contents and form of project report entitled **"TalkFree Mobile Voip Application"** submitted by 1) Summaya Mumtaz, 2) Shajiha Javed, and 3) Captain Yasir Hussain have been found satisfactory for the requirement of the degree.

Supervisor: _____

Dr. Asif Masud

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our parents, without whose unflinching support and unstinting cooperation, a work of this magnitude would not have been possible

ACKNOWLEDEMENTS

We are eternally grateful to Almighty Allah for bestowing us with the strength and resolve to undertake and complete the project.

We gratefully recognize the continuous supervision and motivation provided to us by our Project Supervisors, Dr. Asif Masud fom MCS. We are highly gratified to our committee members Asst Prof. Dr Awais Majeed, Lec Hammad Afzal and Lec Bilal Raoof for their continuous and valuable suggestions, guidance, and commitment. We are highly thankful to all of our professors whom had been guiding and supporting us throughout our course and research work. Their knowledge, guidance and training enabled us to carry out this research work.

We would like to offer our admiration to all our classmates, and our seniors who had been supporting, helping and encouraging us throughout our thesis project. We are also indebted to the MCS system administration for their help and support.

We deeply treasure the unparallel support and tolerance that we received from our friends for their useful suggestions that helped us in completion of this project.

In the end we are also deeply obliged to our families for their never ending patience and support for our mental peace and to our parents for the strength that they gave us through their prayers.

Table of Contents

Chapter 1: INTRODUCTION……5

1.1 Introduction……5

1.2 Background……8

1.3 Problems Addressed……12

1.4 Goals and Objectives……12

1.5 Deliverables……..13

Chapter 2: LITERATURE REVIEW……...14

2.1 Existing Systems……...14

2.1.1 Sipdroid……14

2.1.2 Viber……15

2.1.3 Fringe……....15

2.2 Proposed System……...16

Chapter 3: SYSTEM REQUIREMENT SPECIFICATION……..17

3.1 Functional Requirements for Client……....17

3.2 Functional Requirements for Server……...20

3.3 Non-functional Requirements……22

3.3.1 Performance Requirements ……..22

3.3.2 Security Requirements……..22

3.3.3 Software Quality Attributes……..22

Chapter 4: SYSTEM DESIGN……23

4.1 Assumptions and Dependencies……23

4.2 General Constraints…….23

4.3 Design Pattern…….23

4.4 Use-Case Diagram & Description……25
4.5 System Architecture Details & Diagrams……....36

Chapter 5: IMPLEMENATION……..……52

5.1 Introduction……...52
5.2 Implementation……..53
5.3 Implementation Language……54
5.4 Distribution of Modules……54
5.5 Distribution of Classes with respect to modules……...54
5.5.1 Server Module……55
5.5.2 Client Module……58
5.6 Deliverables……...62
5.7 Summary……....62

CHAPTER 6: TESTING……..63

6.1 Unit Testing……..63

6.2 Integration Testing……80

CHAPTER 7: USER MANUAL……..……..81

7.1 System Requirements……81

7.2 Installation……..…81

7.2.1 Server Installation……81

7.2.2 Client Installation……81

7.3 How to use software……81

CHAPTER 8: CONCLUSION & FURTUE WORK......84

REFERENCES……85

CHAPTER # 01 : INTRODUCTION

1.1 INRODUCTION:-

Imagine being able to call your relatives in USA for the cost of a local phone call. Imagine being able to connect to all of your relatives and friends all over the world free of cost or as low as you make any local call. Imagine having your friends call you free from anywhere in the world using the internet.

All of these applications are now possible thanks to advances in VoIP also referred as internet telephony or IP telephony technology. The great thing about VoIP is that it taps additional value from the already existing infrastructure without additional costs. VoIP transmits the sounds you make over the standard Internet infrastructure, using the IP Protocol. This is how you can communicate without paying for more than your monthly Internet bill. And this is the main reason for which people are so massively turning to VoIP technology. VoIP is said to be cheap, but most people use it for free. The key to successful use of VoIP revolves around making internet telephony invisible; in other words, just like an ordinary phone call.

The good news is that advances in modem technology and in the internet itself, have made internet telephony practical on mobile phones. But there are also advancements in the mobile phone devices and every day we have new mobile devices with new operating system, new capabilities and new features. One of such latest mobile devices includes Android mobile phones. Android has become one of the most popular operating systems for mobile devices and Android phones continue to gain market share over popular smartphones like Apple's iPhone and RIM's Blackberry because of which several new additions to the Android family have been announced. As its era of VoIP and Android mobile devices hence lot of mobile companies and software houses are developing new applications for Android mobile phones.

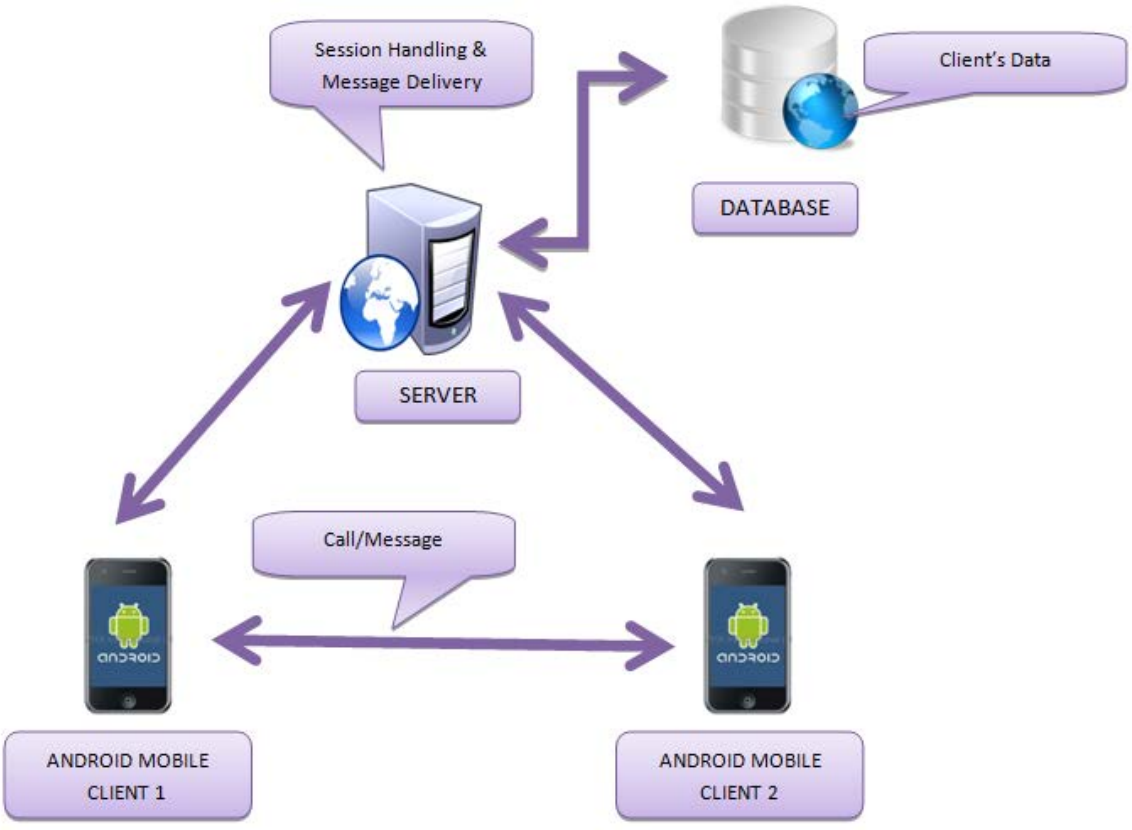
Keeping in view the future demand of VoIP application for Android mobile phones this project aims at developing invisible VoIP application for Android mobile phones. Application will allow users to make calls over internet like any ordinary phone call. The username of each user will be actually mobile number of the user. All the users registered and using this application will have their mobile numbers as their usernames. Hence you don't need to ask or remember usernames of your relatives and friends for making calls over internet. If you know mobile number of your friends or relatives you can add to application contact list and make calls over internet as you make ordinary calls from your mobile phone. Application will allow user to make, hold and disconnect audio calls. Application will store username, password and contact list of user in database. Furthermore application will use compression technique to enhance the quality of service.

1.2 BACKGROUND:-

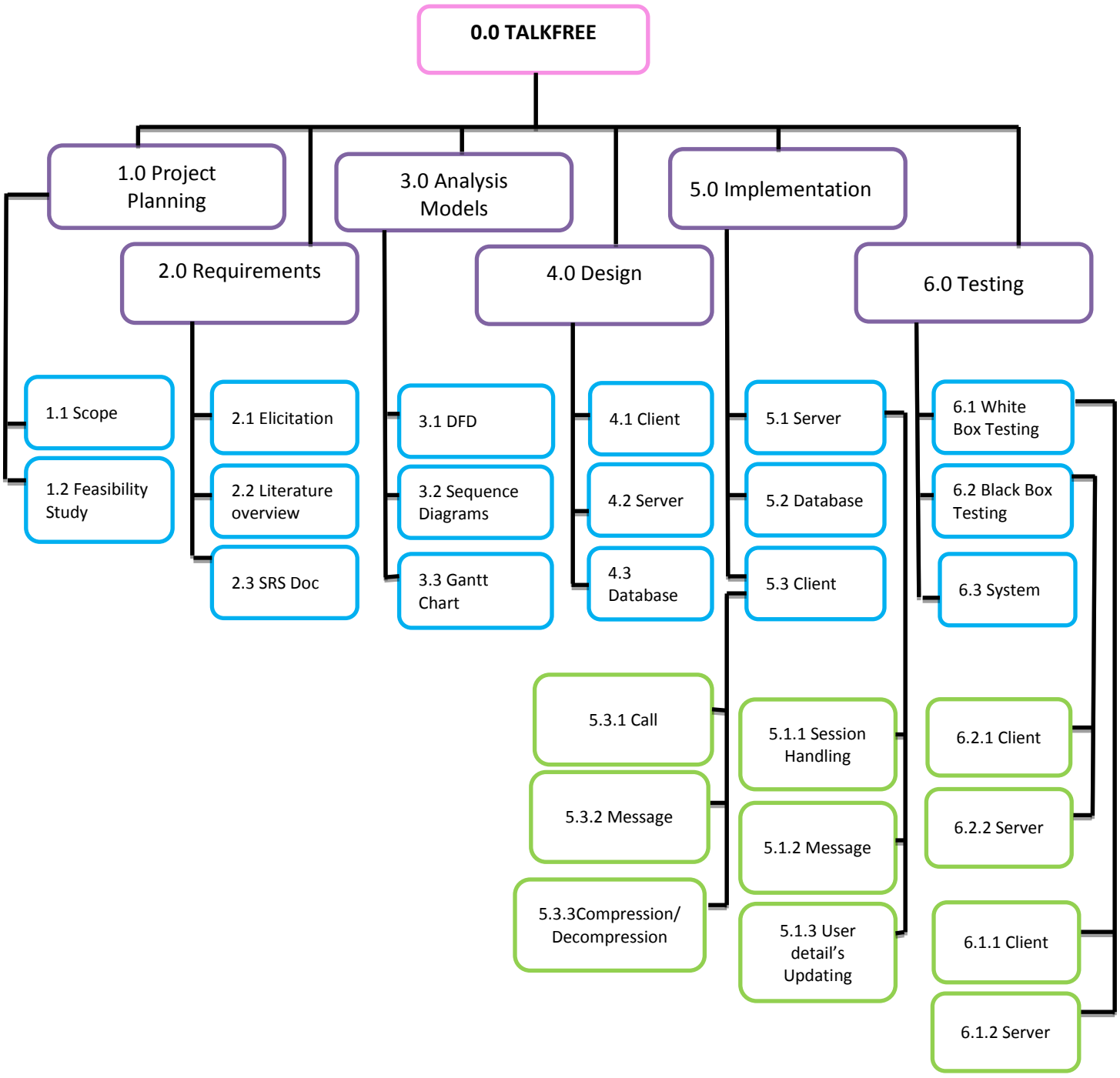
The project aims at developing mobile application that will enable mobile users to call other users via internet. Application will allow user to make audio calls, text messages, conference call, hold or disconnect calls. Application will enable user to send message to other users. The project also aims at compressing the audio to provide better quality service to the users over the congested network.

System is composed of following three modules

- **CLIENT:** Client module is Android based mobile application that will allow user to call via internet. Client module consists of the functionality of registration , login , call , conference call , end call , manage contact list and sending message or to other users
- **SERVER:** Server handles all calls. Server initiates session between users and stores user information in the database. Sever also handles message sending.
- **DATABASE:** Database consists of each user's information i.e. ID, password and contact list



WORK BREAKDOWN STRUCTURE:-

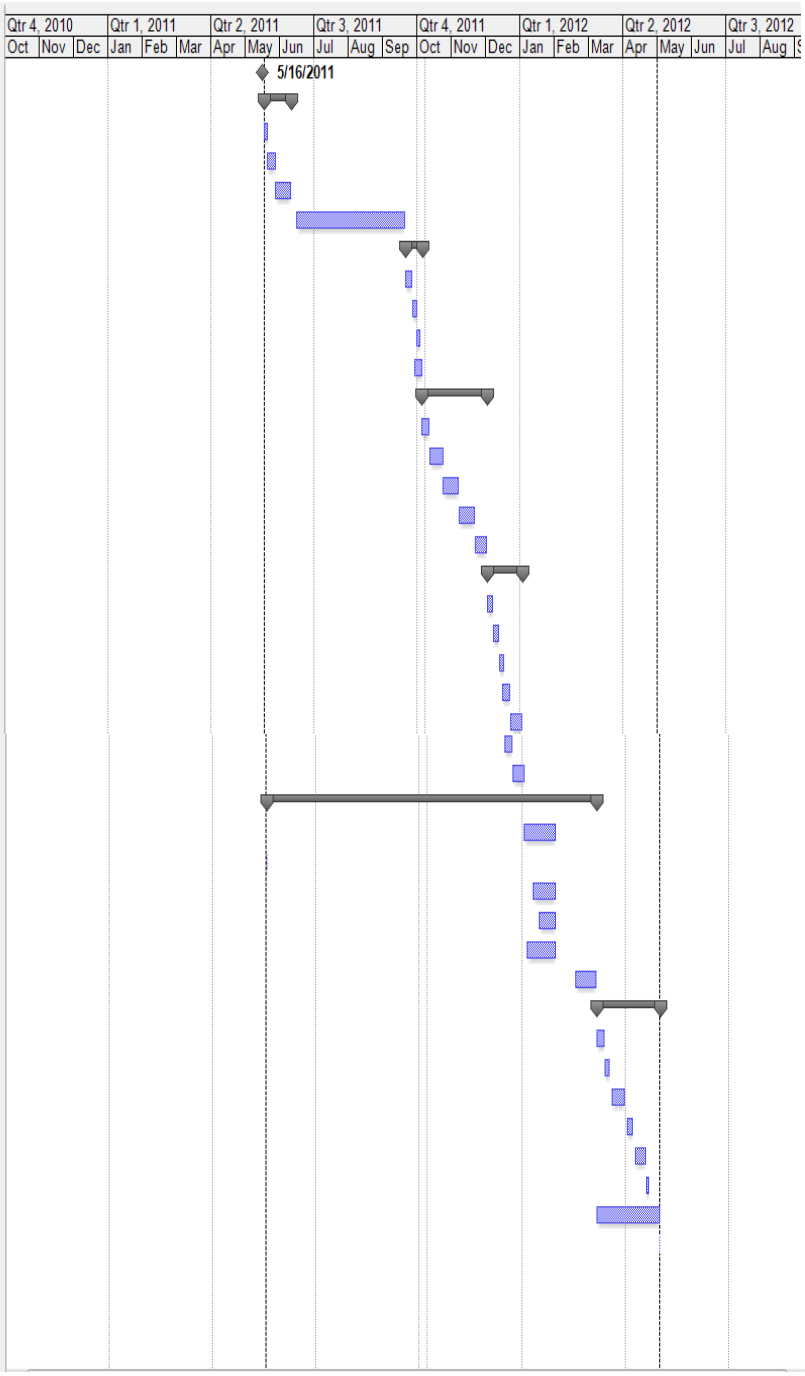


ROLES & RESPONSIBILITIES MATRIX

Name	Responsibility
Member 1: Summaya Mumtaz	-Defence Report -Detailed Analysis of the project -Requirement Specification Document -Architectural Design -Detailed Design -Implementation -Testing -Final Project Report
Member 2: Shajiha Javed	-Defence Report -Detailed Analysis of the project -Requirement Specification Document -Architectural Design -Detailed Design -Implementation -Testing -Final Project Report
Member 3: Yasir Hussain	-Interview with different Stakeholders of the project -Review of latest technologies in the market -Report on technologies -Architectural Design -Detailed Design -Implementation -Testing -Final Project Report

PROJECT PLAN (TIMELINE & MILESTONES)

	Task name	Start	Finish
1	Start of project	5/16/2011	4/30/2012
2	**Project Plannig**	**5/17/2011**	**6/10/2011**
3	Estimation	5/17/2011	5/20/2011
4	Scope	5/20/2011	5/27/2011
5	Feasibility	5/27/2011	6/10/2011
6	Study of existing ...	6/15/2011	9/19/2011
7	**Requirement ...**	**9/20/2011**	**10/5/2011**
8	Requirement ...	9/20/2011	9/26/2011
9	Requirement ...	9/27/2011	9/30/2011
10	Requirement ...	9/30/2011	10/3/2011
11	Software Req...	9/28/2011	10/5/2011
12	**Analysis Phase**	**10/5/2011**	**12/1/2011**
13	System Modling	10/5/2011	10/11/2011
14	Context model	10/12/2011	10/24/2011
15	Bahavioural ...	10/24/2011	11/7/2011
16	Structural mo...	11/7/2011	11/21/2011
17	Architectural ...	11/21/2011	12/1/2011
18	**Design Phase**	**12/2/2011**	**1/2/2012**
19	Architectural ...	12/2/2011	12/6/2011
20	Abstract speci...	12/7/2011	12/12/2011
21	Interface Desi...	12/13/2011	12/16/2011
22	Data structur...	12/16/2011	12/22/2011
23	Algorithm desi...	12/23/2011	1/2/2012
24	**Implementatio...**	**5/18/2011**	**3/5/2012**
25	Implementatio...	1/2/2012	1/30/2012
26	Implementatio...	5/18/2011	5/18/2011
27	Quality issues...	1/10/2012	1/30/2012
28	Debugging	1/16/2012	1/30/2012
29	Performance ...	1/5/2012	1/30/2012
30	Test(unit testi...	2/16/2012	3/5/2012
31	**Testing Phase**	**3/6/2012**	**4/30/2012**
32	Code Inspection	3/6/2012	3/12/2012
33	Client Applicat...	3/13/2012	3/16/2012
34	Server Testing	3/19/2012	3/30/2012
35	Database test...	4/2/2012	4/6/2012
36	Integration te...	4/9/2012	4/18/2012
37	System Testing	4/19/2012	4/20/2012
38	Regression te...	3/6/2012	4/30/2012
39	Final report	4/30/2012	4/30/2012



1.3 PROBLEMS ADDRESSED:-

Application has addressed following problems successfully

- Login
- Call establishment using Sip protocol
- Sending and receiving messages from other user
- Compressing audio packets using G711 algorithm
- Managing contact list of user , allowing adding and deleting contacts

1.4 GOALS & OBJECTIVES:-

» **Academic Objectives :** This project aims at achieving following academic objectives

- Understand Android operating system and develop application for Android OS
- Study various compression techniques and application of best compression technique
- Integration of database with application
- Application of internet telephony for mobile phones
- Understanding and application of client server architecture
- Development of applications for mobile devices
- Understanding of network programming (NAT traversal techniques)
- Develop application following phases of software development life cycle
- Applying strategies (studied so far) on different phases of SDLC

» **Application/End Goal Objectives :** To provide

- Audio mobile call service via internet for Android operating system. The ID of the users will be their mobile number that is unique for every user. To call any other user, client will only dial mobile number of other user. Application will work like ordinary phone calls.
- Hold or disconnect call feature.
- Message feature that will enable user to send message to any other user.
- Better quality communication over internet using compression scheme. All voice data before going on internet will be compressed

1.5 DELIVERABLES:-

- » Defense Report
- » Literature Review & Software Requirement Specification Document
- » Architectural Design Report
- » Detailed Design Report
- » Software System
  - Client Module
  - Server Module
  - Database
  - Integrated System

- » Testing Report
  - Client Module testing document
  - Server Module testing document
  - Integrated System testing document

- » Final Report

CHAPTER # 02 : LITERATURE OVERVIEW

2.1 EXISTING SYSTEMS

2.1.1 SIPDROID:-

Android SIP client application sipdroid enables customers to make free phone calls to other VoIP users or very cheap phone calls to anyone else in the world from your mobile phone. Sipdroid presently supports basic call. All the typical PBX features are configured from the PC. This concept is much like the concept of Google's services that allow you e.g. to manage your calendar on the PC and access it remotely on your phone.

» **FEATURES:** The following features are provided by the Free Account

- Support of several modes for DTMF tones
- Support for NAT (network address translation)
- Simultaneous Outbound Calling
- Screening anonymous callers
- Time-based routing for incoming calls
- Attended Call Transfer
- Conferences
- Video Reception (Video Transmission is supported by Sipdroid natively)
- Trigger callback or call thru (if no suitable data network available)
- Calls to Skype users

» **LIMITATIONS:** Sipdroid have following issues

- Audio quality appears to be good for outbound from phone but the user is unable to hear any incoming video call
- Hold-place a call on hold and retrieve (failed to do so)
- Incoming calls are not received when phone is suspended /sleep
- Only register when placing a call
- Search and replace option does not work
- Registration failed when reconnects to a network
- Video call reception no way to stop receiving video
- Does not connect over 2G
- Voice is transferred before a call is received
- Pause and wait not supported
- Heavy background noise
- Installing sipdroid 1.1.5 from the market on the Samsung, configure a SIP account and make a SIP or normal PTSN phone call. There is no audio, send or

received. Even for normal phone calls. As soon as sipdroid is uninstalled, phone calls work normally again.
- On 3G sipdroid tries to connect but gets a timeout answer.

2.1.2 VIBER:-

Viber is an application for Android™ phones that lets you make free phone calls and send text messages to anyone who also has the application installed.

» **FEATURES:** Viber has following features

- You can call or text any Viber user, anywhere in the world, for free.
- Viber integrates seamlessly with your existing contact list. As soon as you open the Viber application, Viber will scan your contact list, and if it recognizes another Viber user, it will place a badge next to that person's name. It will do the same in their device. So just like with a regular phone, only the people who already have your phone number can call you.
- Has great sound quality
- Once activated, does not require a PIN, username or any additional "in application" purchase.

» **LIMITATIONS:-**

- Needs a lot of bandwidth for good quality service. The bandwidth rate during a Viber call is approximately 240 KB per minute, 14 MB per hour.
- Viber does not inform you of who is online or not. If the party you are calling doesn't have an active internet connection or has their Viber application turned off when you call, the call will time out

2.1.3 FRING:-

Fring is a peer-to-peer mobile VoIP internet telephony application. Fring allows making free mobile calls, video calls and living chat to friends.

» **FEATURES** : Fring has following features

- Fring supports four way live video calling
- Mobile calls
- Live chat with friends added in your buddy list
- Buddy list : Search and add friends to your buddy list
- Fring enables user to visit fiends' profiles
- Invite friends to join fring network
- Maintains call history/activity log
- **Personalize your profile:** upload a picture & edit mood, email address and phone number.

» **LIMITATIONS**

- With more fring user's video calls their friends on Android, Nokia and iPhones; there is network 'stress' that affects quality of service.
- Many times fring login fails
- On some of the handsets fring call facility is not working giving error "unable to make call"
- Audio input sample rate control has some bugs because of which the audio shuts off upon connection when making a call.

2.2 PROPOSED SYSTEM:-

Proposed system will cater for some of the limitations of the applications mentioned above.

- Main emphasizes is to make invisible audio call application with no issues like login error or poor voice quality.
- Application will enable user to send message
- Compression techniques will be used to increase quality of service.

CHAPTER 3: System Requirements Specifications

3.1 Functional Requirements for Client

3.1.1 Login

- **Description**

The system allows user to login when the user will start application for the first time.

- **Stimulus/Response Sequences**

The application will allow user to enter his username based on the mobile number and password. Sever will check if username and password exists in the database and username and password matches , user will be able to successfully login and home screen will be displayed to the user

- **Functional Requirements**

REQ-1: Login to the system

3.1.2 Call users

- **Description**

Application will display list of users and user can select any contact and click on call button to call that user.

- **Stimulus/Response Sequences**

After login to the system the application will be displaying contacts in the contact list. User will select any contact and then click on call button. User will be listening ringing bell. Server will establish session between them . After successful connection users will be able to talk with each other.

- **Functional Requirements**

REQ-1: Display list of users in the contact book.

REQ-2: Select any contact in the contact book.

REQ-3: Call selected contact.

REQ-4: Bell ringing before the call is received at the other end.

3.1.3 Send Message

- **Description**

User can select any user and send message .

- **Stimulus/Response Sequences**

Application will display icon of message against the name of each contact in the contact list. User will click on the message icon and application will display next screen enabling user to write text send to other user. User will get notification according to delivery status of the message.

- **Functional Requirements**

REQ-1: Send message.

REQ-2: Display notification

3.1.4 Add Contact:-

- **Description**

This feature will enable user to add contacts that have registered accounts. User will enter name and mobile number . If that mobile number is found in the database, that contact will be added to the contact list of the user.

- **Stimulus/Response Sequences**

Application will provide “Add Contact” button at the end of the contact list. User will click on add contact button. Next screen will allow user to enter name and mobile number . After entering name and mobile number user will click “Add” button. Server will search that mobile number in the database. If the number exists in the database, it will be added to the contact list of the user. If number is not found message will be displayed “User not found’

- **Functional Requirements**

REQ-1: Enter name and mobile number of other user.

REQ-2: Search database for the number entered by the user.

REQ-3: If number found add to the contact list of the user.

REQ-4: If number not found, display message and screen to enter name and number again.

3.1.5 Hold call

- **Description**

Application will enable user to hold/unhold a call.

- **Stimulus/Response Sequences**

When user presses “hold” button the call be placed on hold and when user presses “unhold” button call will be active again.

- **Functional Requirements**

REQ-1: Hold call

REQ-2: Unhold call

3.1.6 Disconnect call

- **Description**

Application will enable user to disconnect call.

- **Stimulus/Response Sequences**

When user presses “disconnect”, call will be ended.

- **Functional Requirements**

REQ-1: Disconnect call.

3.1.8 Compress/Decompress Data

- **Description**

All application data will be compressed and decompressed for improving performance and quality of the voice.

- **Stimulus/Response Sequences**

Application will compress voice data before sending over the internet and then decompress incoming data.

- **Functional Requirements**

REQ-1: Compress outgoing data

REQ-2: Decompress incoming data

3.2 System Features for Server

3.2.1 Register User

- **Description**

Server will store username and password entered by the user in the database.

- **Stimulus/Response Sequences**

Server will receive username and password from the client applications and check it in the database. If the username is unique, server will add username and password in the database and send account registration confirmation message to client application.

- **Functional Requirements**

REQ-1: Check username entered by the client doesn’t exist in the database

REQ-2: Add username and password in the database record

REQ-3: Send account registration confirmation message to client

3.2.2 Add contacts

- **Description**

Client will send contact number to be added in the contact list. Server will check contact number sent by the user ,exists in the database. If the contact number exists in the database, contact number and name will be added to the contact list of the user in the database.

- **Stimulus/Response Sequences**

Client will send contact number to the server. Server will check contact number exists in the database. If contact number exists in the database server will add this contact number to the list of contacts in the database and send confirmation message to the user that contact is added to the contact list.

- **Functional Requirements**

REQ-1: Check contact is registered

REQ-2: Add to contact list of user

REQ-3: Notify user

3.2.3 Establish call session between two users

- **Description**

Client will request to call a particular online contact. Server will establish session between both users.

- **Stimulus/Response Sequences:-**

Client will send request to call another online user. Server will forward call request to other user and send acknowledgment to client.

- **Functional Requirements:-**

REQ-1: Send call request to other user

REQ-2: Send ack to client

3.2.4 Store & Send Message

- **Description**

Server will store message sent by the user and will deliver it to other user when the user will come online.

- **Stimulus/Response Sequence**

User will send message to server. Server will check status of other user. If user is online message will be delivered otherwise server will store message and when other user will come online ,server will deliver stored message to that user and notify the sender that message is delivered.

- **Functional Requirements:-**

REQ-1: Check user status

REQ-2: Send message

REQ-3: Notify sender

3.3 Nonfunctional Requirements

3.3.1 Performance Requirements

- Compression algorithm is being implemented in the system which will help to transfer the data quickly and increase the quality of voice

3.3.2 Security Requirements

- To ensure the security of the system the intended user will be authorized to use the system by verifying him through a database and user will have to login if he has already signed up for the system.

3.3.3 Software Quality Attributes

- **Reliability**

  The system is to be used by the different users. So system is needed to be very reliable.

- **Maintainability**

  The system should be maintainable so that in case of any complaints from the users, the system should be modified to meet the new requirements.

- **Availability**

  It should be available to mobile users all the time.

- **Correctness**

  It should provide correct data.

- **Portability**

  It should be portable enough to be moved to different android mobiles.

DESIGN CONSIDERATIONS

4.1 Assumptions & Dependencies

- The Android Mobile Client Application should be connected to the Internet.
- The server application should be running on Web Server.
- The Database should contain all contact information of all the users who use the system.
- All users must be registered on the system to be able to use it.

4.2 General Constraints:-

- This application is developed to be used over internet which may affect the quality of voice.
- The project has to be finished within semester timings hence additional features will be implemented depending upon time.
- To meet deadline of the project, open source SIP server will be used.

4.3 Design PATTERNS:-

- **Information Expert:-**

The Information Expert pattern is a solution to the problem of determining which object should assume a particular responsibility.
During design, we make choices about the assignment of responsibilities to classes. Information Expert helps us decide, once we know the task (responsibility), which class to make responsible for carrying out the task.Information Expert says:

Assign a responsibility to the information expert; the class that has the information necessary to fulfill the responsibility.

Real-world application usually holds a number of classes that perform hundreds and thousands of actions, so it is essential to assign responsibilities correctly. The principle itself is fairly simple – you should give the responsibilities to the object it has information for.

- **Architectural Design**

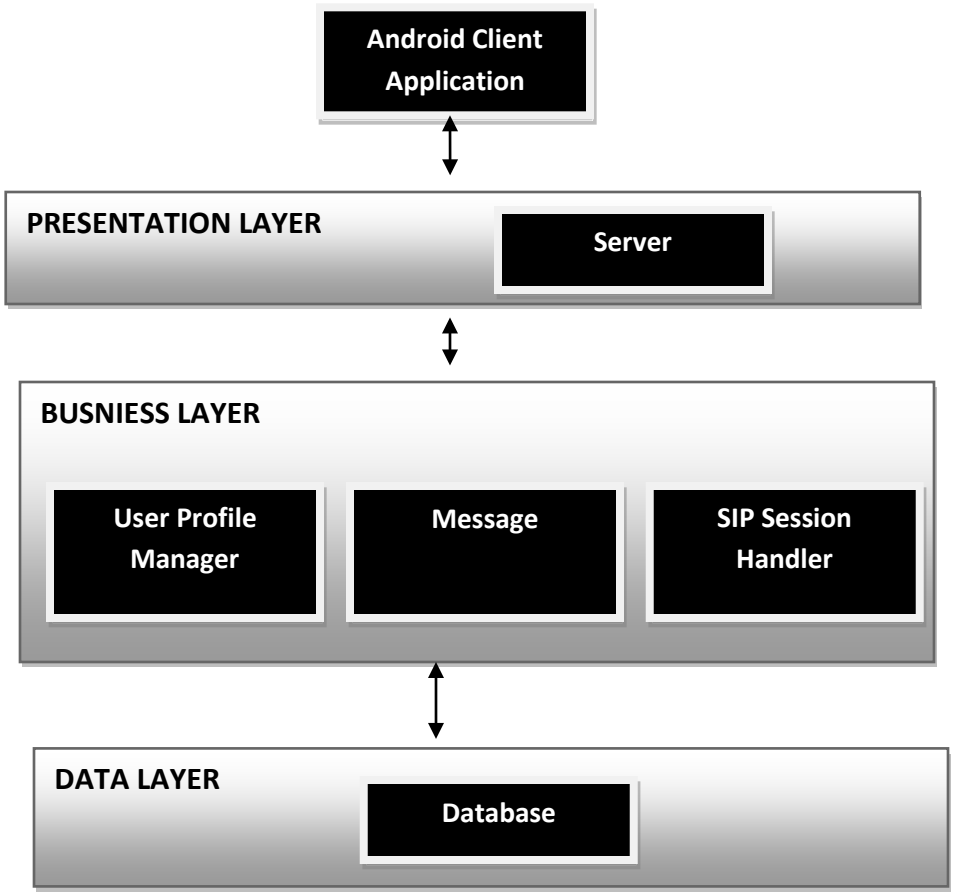
Multi-tier is a client-server architecture in which the presentation, the application processing, and the data management are logically separate processes. The most widespread use of multi-tier architecture is the three-tier architecture, which typically consist of three layers

- ✓ **The data layer**, providing access to the application data

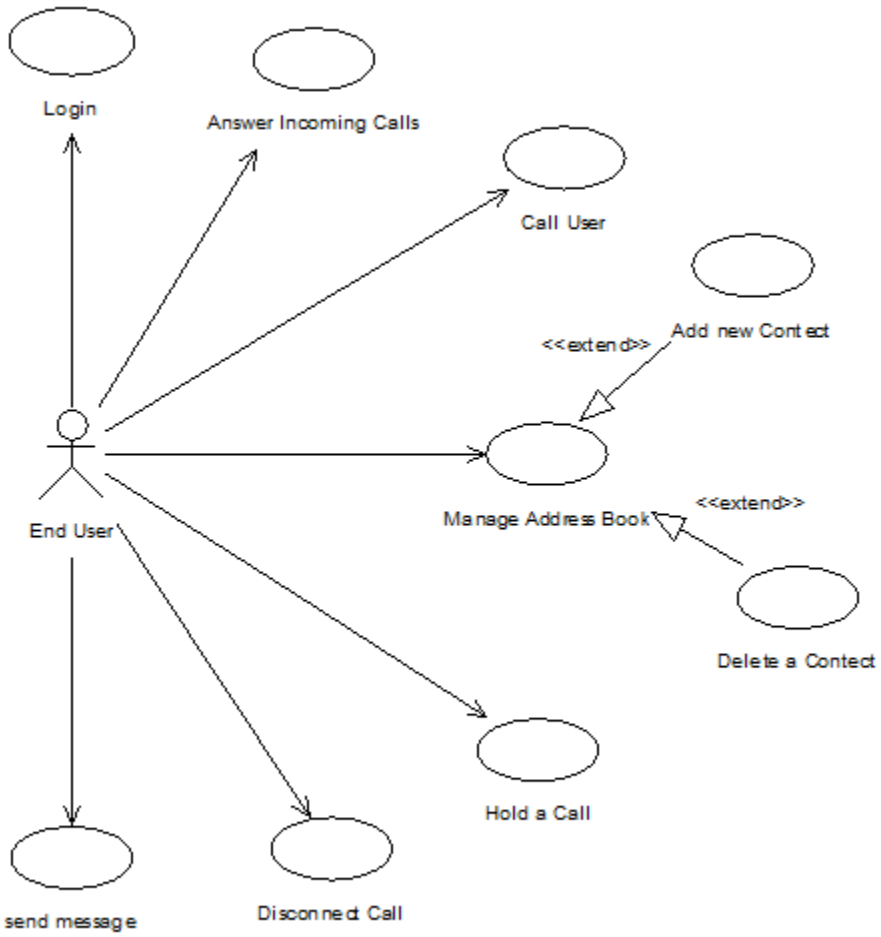
- ✓ **The business layer**, hosting the business logic of the application in an application serve
- ✓ **The presentation layer**, which renders the result of the request in the desired output format.

N-tier application architecture provides a model for developers to create a flexible and reusable application. By breaking up an application into tiers, developers only have to modify or add a specific layer, rather than have to rewrite the entire application over. There should be a presentation tier, a business or data access tier, and a data tier.

Following diagram shows TalkFree n-layer architecture



4.4 Use-case Diagram:-



4.5 Use-case Description:-

- **Login:**

1. Brief Description:

This use case describes how a user login to the application.

2. Actors:

User

3. Pre-conditions:

The application should be active.

4. Basic Flow of Events:

The use case starts when the user wishes to login.

1. The system prompts the user to enter its ID and Password.
2. The user enters the ID and Password.
3. If there is no conflict, user will be login.
4. The use case ends.

5. Alternative Flows:

5.1 Invalid ID:

If, in the Basic Flow, the application determines that the ID entered is not correct or does not exist, an error message is displayed.

5.2 Invalid Password:

If, in the Basic Flow, the application determines that the password entered is not correct, an error message is displayed.

6 Key Scenarios:

N/A

7 Post Conditions:

7.1 Successful Condition:

User is logged in to the application.

7.2 Failure Condition:

User is not logged in.

8 **Special Requirements:** nil

……

- **Manage Contact list:**

1. **Brief Description:**

This use case describes how a user manages its contact list.

2. **Actors:**

User

3. **Pre-conditions:**

The Application should be active and user is logged in.

4. **Basic Flow of Events:**

The use case starts when the user wishes to manage its contacts.

1. The system prompts the user whether he/she wants to add or delete a contact.
2. The user then selects according to his/her requirement.
3. If the user selects to add a new contact, the use case **'Add Contact'** starts with all of its basic flow of events.
4. If the user selects to delete an existing, the use case **'Delete Contact'** starts with all of its basic flow of events.
5. The use case ends.

5. **Alternative Flows:**
N/A

6. **Key Scenarios:**

N/A

7. Post-conditions:

7.1 Successful Condition:

Selected action is performed and the contact list is updated.

7.2 Failure Condition:

Selected action is not performed.

8. Special Requirements:

N/A

……

- **ADD Contact:**

1. Brief Description:

This use case describes how a user adds a new contact in the contact list.

2. Actors:

User

3. Pre-conditions:

The application should be active.

4. Basic Flow of Events:

The use case starts when the user logs in to the application and wishes to add a new contact in his/her contact book.

1. The system prompts the user to enter contact person's name and its mobile no i.e. the unique ID of that person
2. The user enters the name and mobile no.
3. User selects "Add" option.
4. The use case ends.

5. Alternative Flows:

5.1 Invalid ID:

If, in the basic flow, the application determines that the entered ID is invalid or not exists in the database. The application displays an error

message. So at this point the user may enter again or can cancel the operation and the use case ends.

5.2 Invalid Name:

If, in the basic flow, the application determines that the entered name is invalid or already exist in contact list. The application displays an error message. So at this point the user may enter again or can cancel the operation and the use case ends.

6. Key Scenarios:

N/A

7. Post conditions:

7.1 Successful Condition:

Contact is added to the list and updated list is shown.

7.2 Failure Condition:

The application is updated accordingly.

8. Special Requirements:

N/A

.....

- ## Delete Contact:

1. Brief Description:

This use case describes how a user deletes an existing contact from the contact list.

2. Actors:

User

3. Pre-conditions:

The application should be active and some contacts must exist.

4. Basic Flow of Events:

The use case starts when the user logs in to the application and wishes to delete contact from his/her contact list.

1. The user enters the name and mobile no.

1. User selects a contact to remove.

2. User selects “Delete” option.
3. Application displays a confirmation message.
4. User confirms the action.
5. The use case ends.

5. Alternative Flows:

5.1 Database Error:

If, in the basic flow, the application determines that user selected user cannot be deleted from the database due to some error. The application displays an error message. So at this point the user may wait, or can cancel the operation and the use case ends.

6 Key Scenarios:

N/A

7 Post conditions:

7.1 Successful Condition:

Contact is deleted from the list and updated list is shown.

7.2 Failure Condition:

The application is updated accordingly.

8 Special Requirements:

N/A

.....

6. Call User:

1. Brief Description:

This use case describes how a user calls an online user.

2. Actors:

User

3. Pre-conditions:

The application should be active and some contacts must be exists in contact list.

4. Basic Flow of Events:

The use case starts when the user logs in to the application and wishes to call another user from his/her contact list.

1. The application display contact list.
2. The user selects a contact to call.
3. User selects “Call” option.
4. The use case ends.

5. Alternative Flows:

5.1 Call to an offline contact:

If, in the basic flow, the application determines that the selected contact is offline. The application displays a message. So at this point user can cancel the operation and the use case ends.

5.2 Network Problem:

If, in the basic flow, the application determines that the network is down. The application displays a message. So at this point the user may try again later or can cancel the operation and the use case ends.

6 Key Scenarios:

N/A

7 Post conditions:

7.1 Successful Condition:

Call is successfully established.

7.2 Failure Condition:

The application is updated accordingly.

8 Special Requirements:

N/A

.....

7. Answer Incoming Call:

1. Brief Description:

This use case describes how a user answers an incoming call.

2. Actors:

User

3. Pre-conditions:

The application should be active and user must have an incoming call.

4. Basic Flow of Events:

The use case starts when the user has an incoming call.

1. User Select answer option to attend the call.
2. The use case ends.

5. Alternative Flows:

5.1 Call is rejected:

If, in the basic flow, the application determines that the user rejected the call. The application displays a message.

5.2 No Answer:

If, in the basic flow, the application determines that the user is not answering the call. The application ends the call and use case ends.

6 Key Scenarios:

N/A

7 Post conditions:

7.1 Successful Condition:

User answers the call and call is connected.

8 Special Requirements:

N/A

.....

8. Send Message:

1. Brief Description:

This use case describes how a user sends a message to a particular contact.

2. Actors:

User

3. Pre-conditions:

The application should be active and a contact is selected.

4. Basic Flow of Events:

The use case starts when the user wishes to send a message to a contact added in his/her contact list.

1. Application prompts for entering the message.
3. User type the text message.
4. User selects “Send” to send the message.

5. Alternative Flows:

5.1 Network Problem:

If, in the basic flow, the application determines that the network is down. The application displays a message. So at this point the user may try again later or can cancel the operation and the use case ends.

5.2 Send an empty message:

If, in the basic flow, the application determines that the user is trying to send without typing any text. The application displays error message. So that at this point user may enter text again or can cancel the operation and use case ends.

6 Key Scenarios:

N/A

7 Post conditions:

7.1 Successful Condition:

Entered message is sends to the selected contact.

8 Special Requirements:

N/A

.....

9. Disconnect Call:

1. Brief Description:

This use case describes how a user ends a call.

2. Actors:

User

3. Pre-conditions:

The application should be active and call is established between user and any other contact.

4. Basic Flow of Events:

The use case starts when the user wishes to disconnect a call.

1. User selects “Disconnect Call”.
2. Application ends the call.
5. Use case ends.

5. Alternative Flows:

N/A

6. Key Scenarios:

N/A

7. Post conditions:

7.1 Successful Condition:

Call ends successfully.

8 Special Requirements:

N/A

.....

10. Hold a Call:

1. Brief Description:

This use case describes how a user holds a call.

2. Actors:

User

3. Pre-conditions:

The application should be active and call is established between user and any other contact.

4. Basic Flow of Events:

The use case starts when the user wishes to disconnect a call.

1. User selects “Hold Call”.
2. Application holds the call.
5. Use case ends.

5. Alternative Flows:

N/A

6. Key Scenarios:

N/A

7. Post conditions:

7.1 Successful Condition:

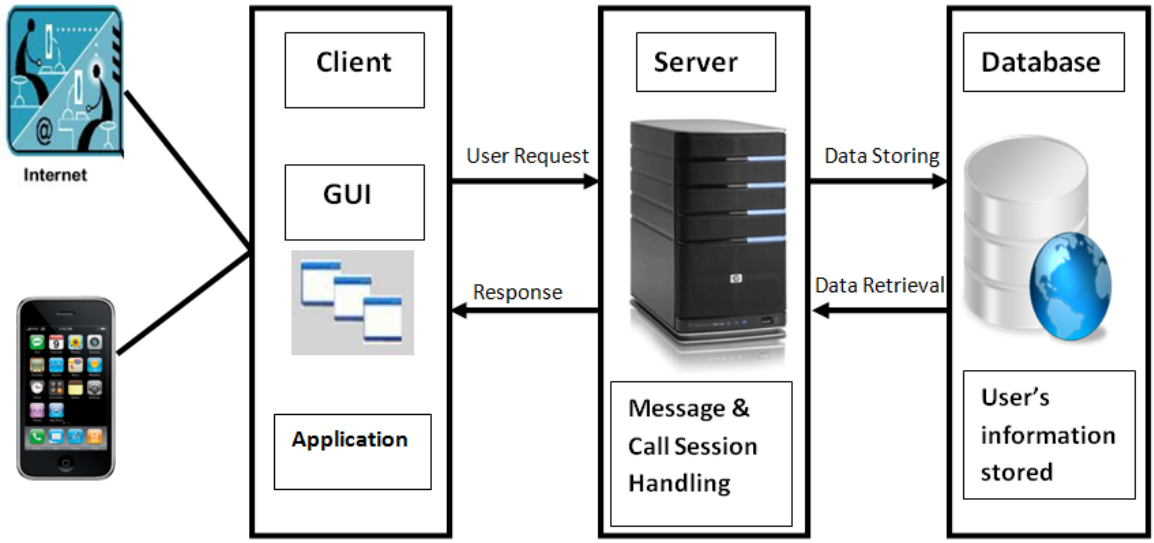
Call hold successfully.

8. Special Requirements:

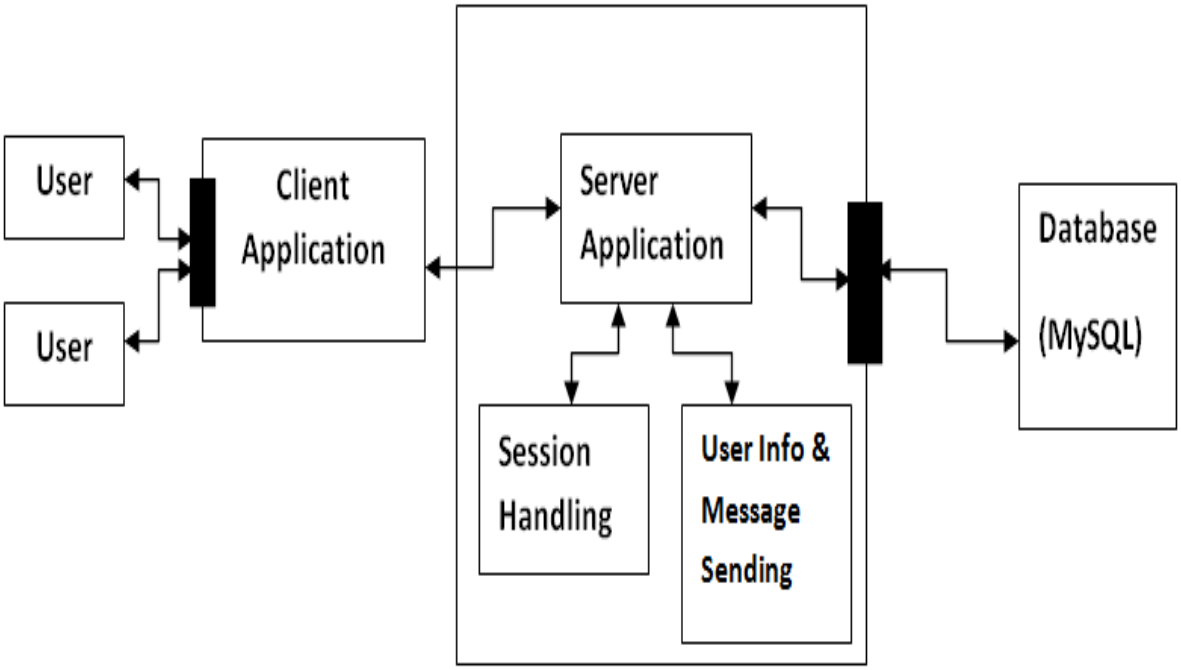
N/A

SOFTWARE DESIGN

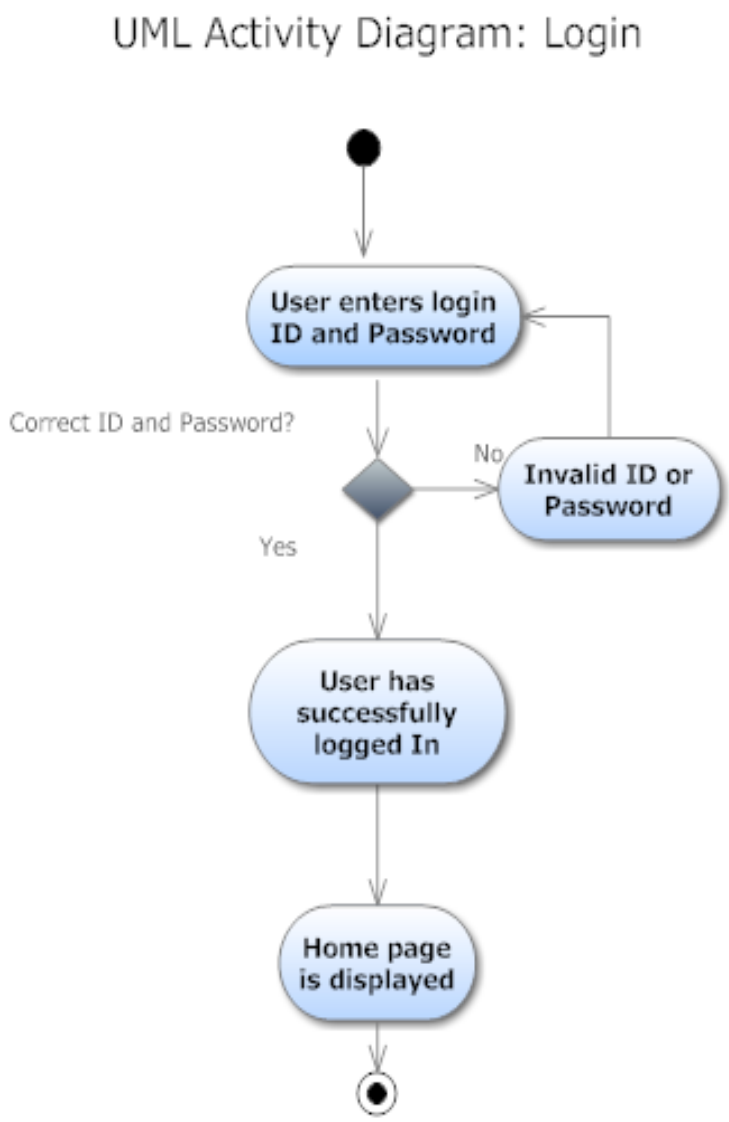
3.1) System Architecture:-

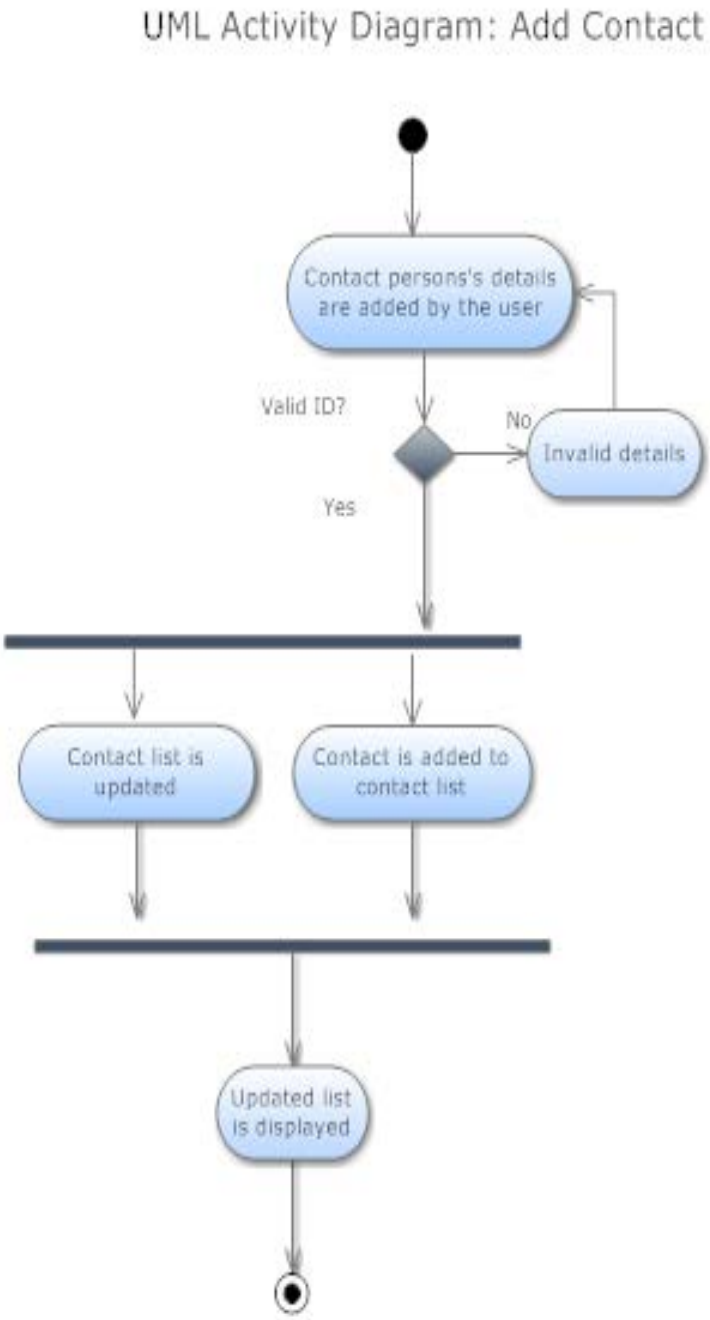


- **Architecture Design**



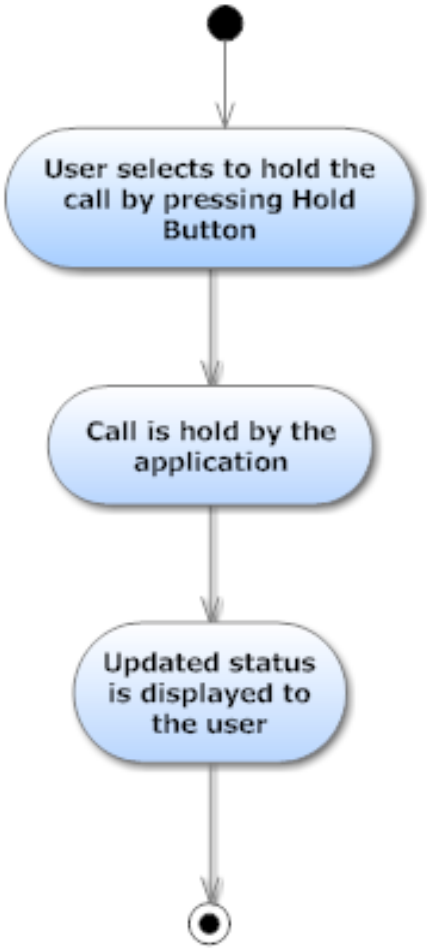
3.2) Activity Diagram:-



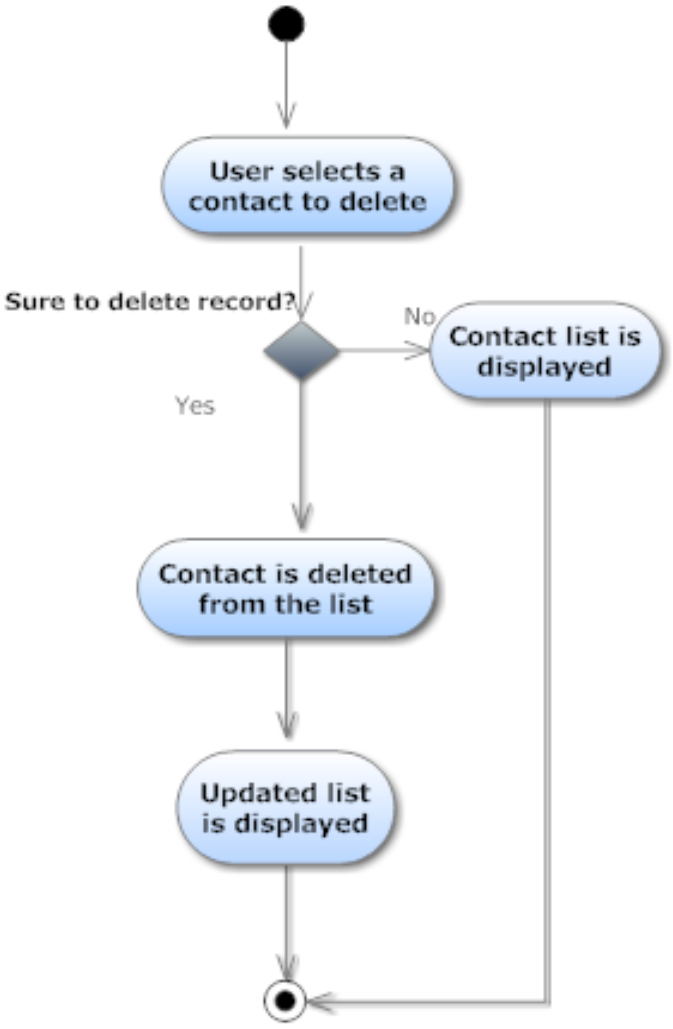

UML Activity Diagram: Add Contact
Contact persons's details are added by the user
Valid ID?
No
Invalid details
Yes
Contact list is updated
Contact is added to contact list
Updated list is displayed

UML Activity Diagram: Call Disconnect

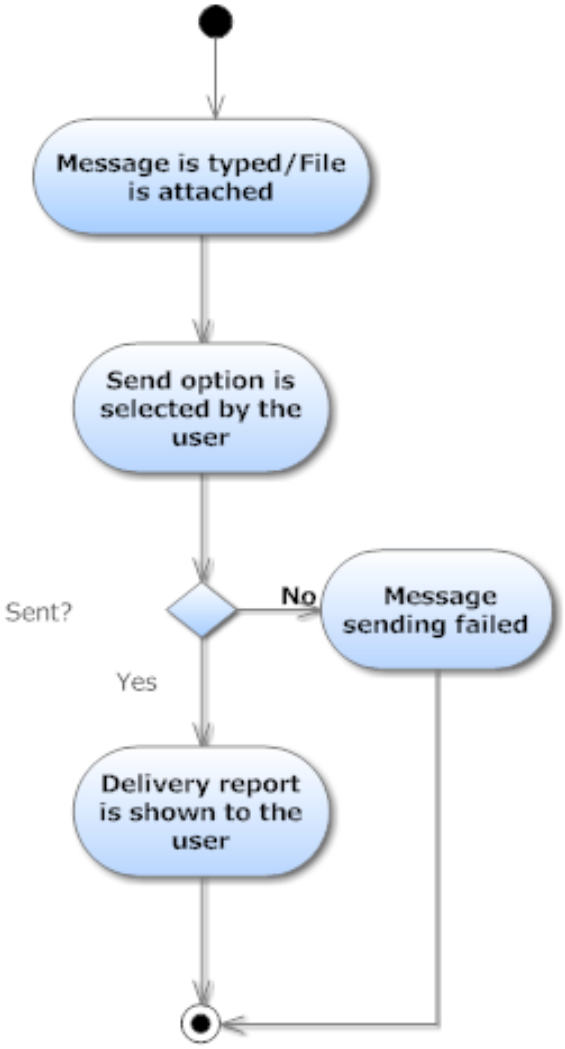
UML Activity Diagram: Call Hold



UML Activity Diagram: Delete Contact



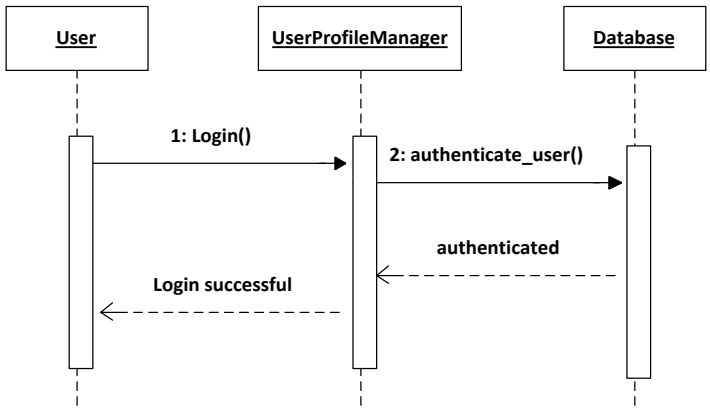
UML Activity Diagram: Send Message



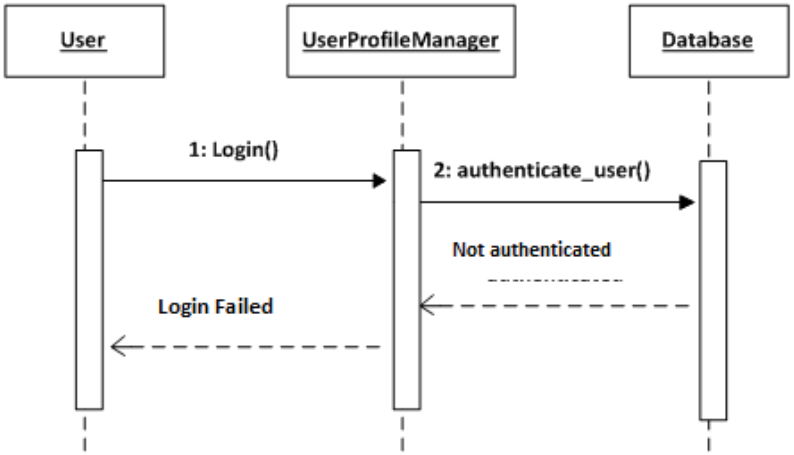
3.3) Sequence Diagram:-

- **LOGIN**

  - **SUCCESSFUL FLOW**

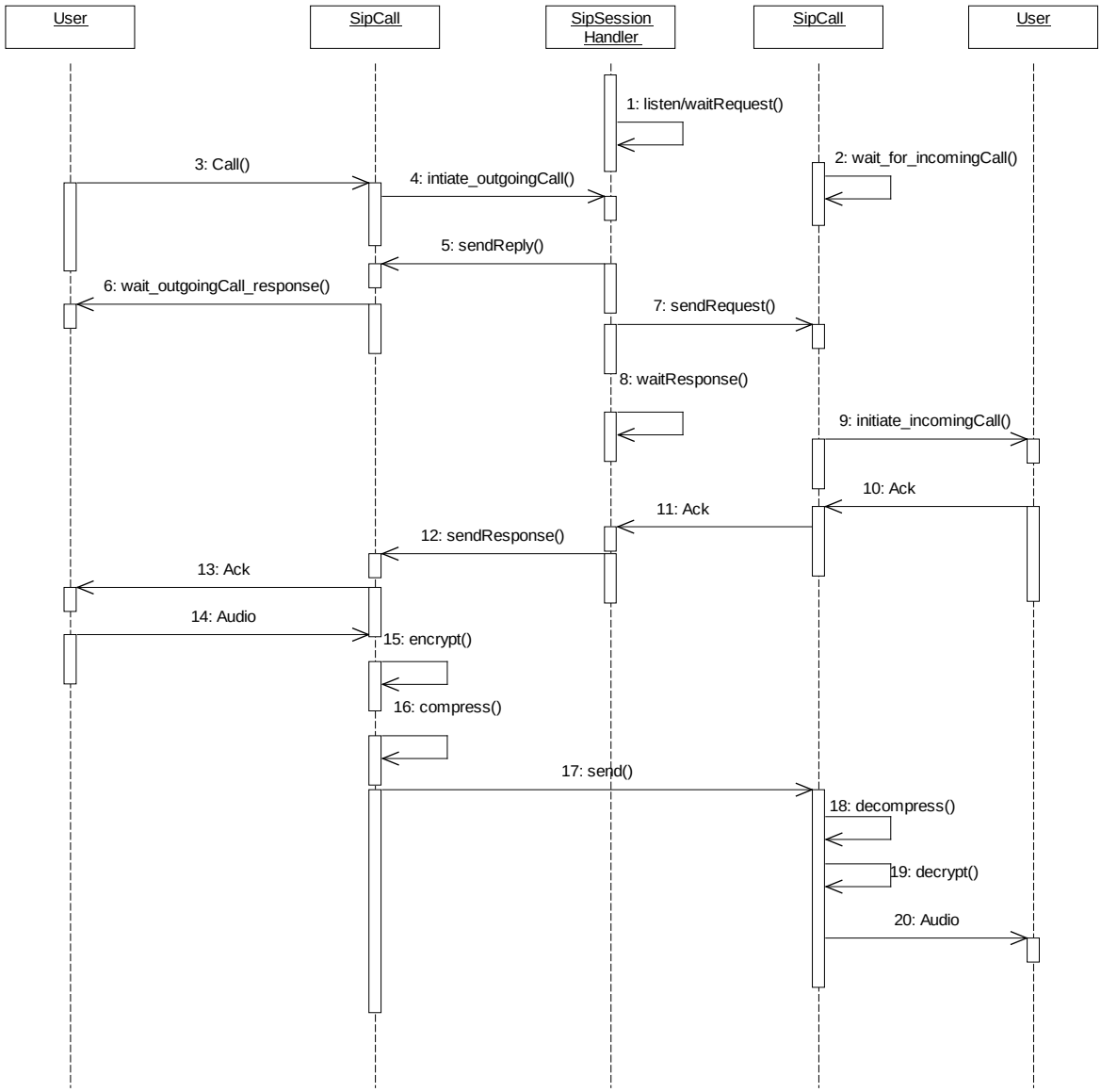


- **ALTERNATE FLOW**

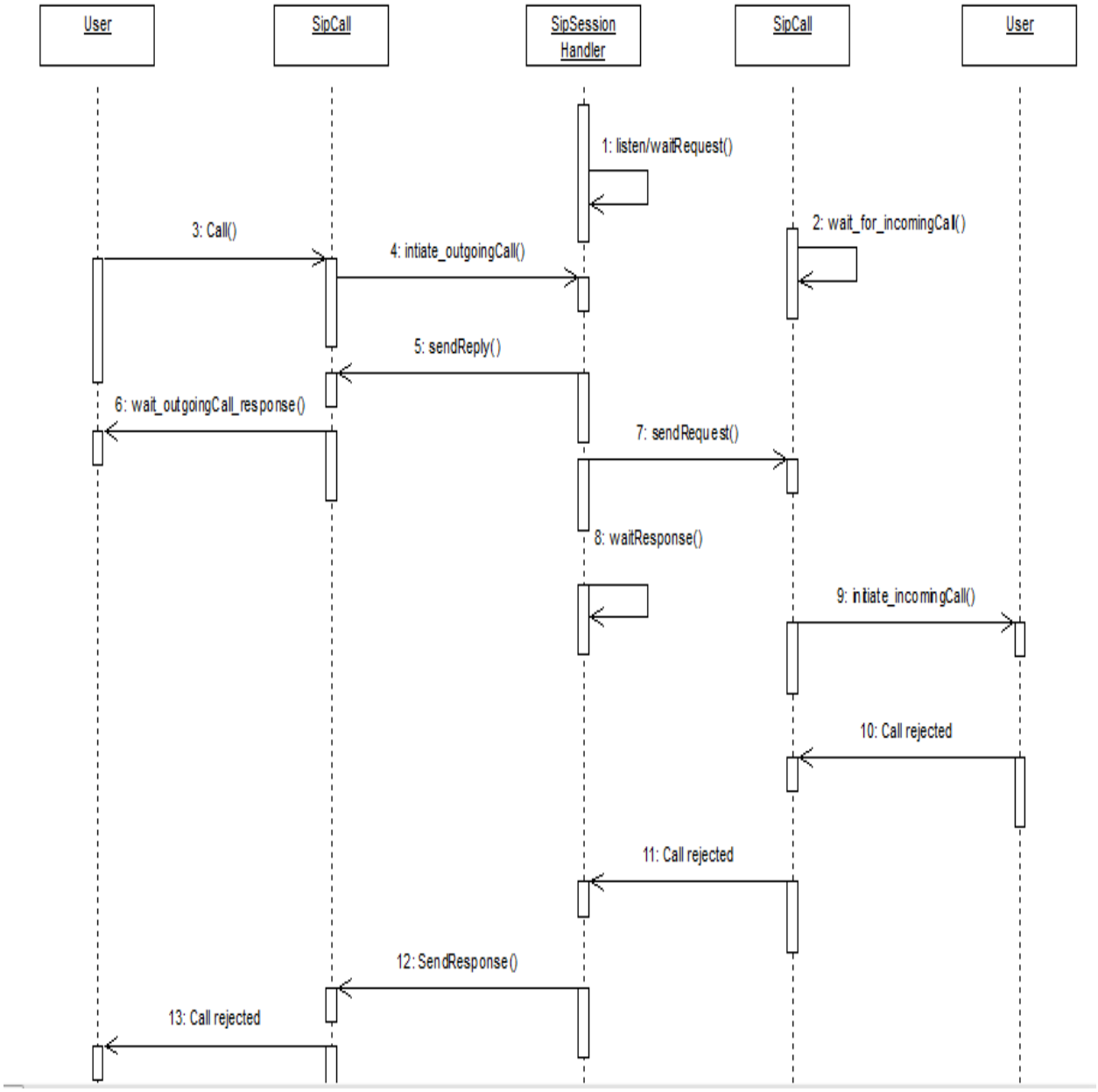


o CALL

- SUCCESSFUL FLOW

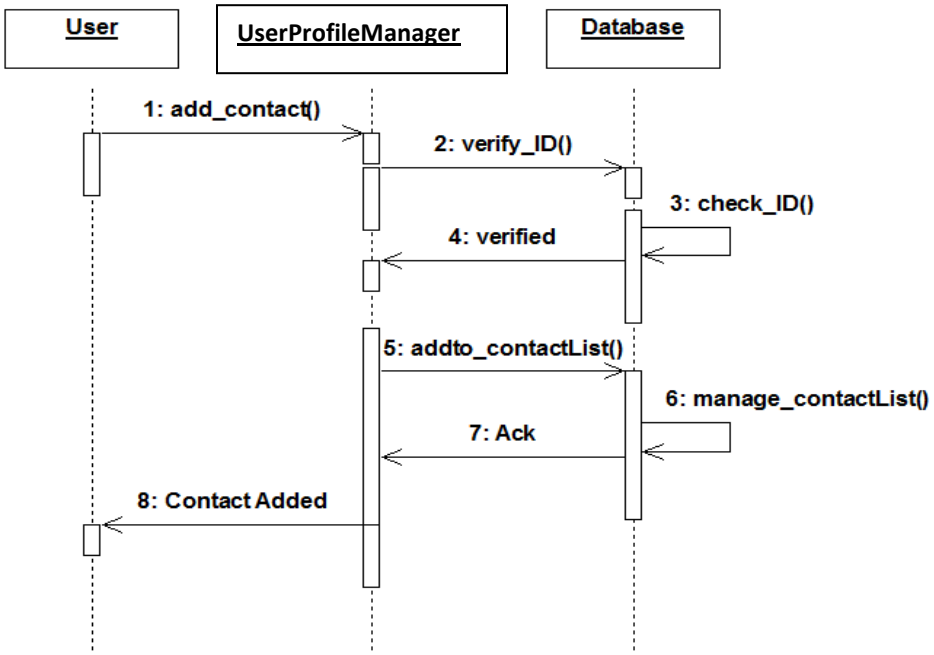


- **ALTERNATE FLOW**

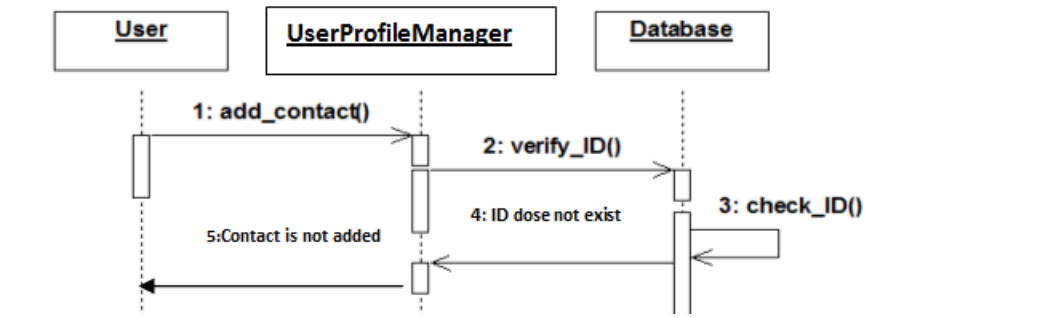


o **ADD CONTACT**

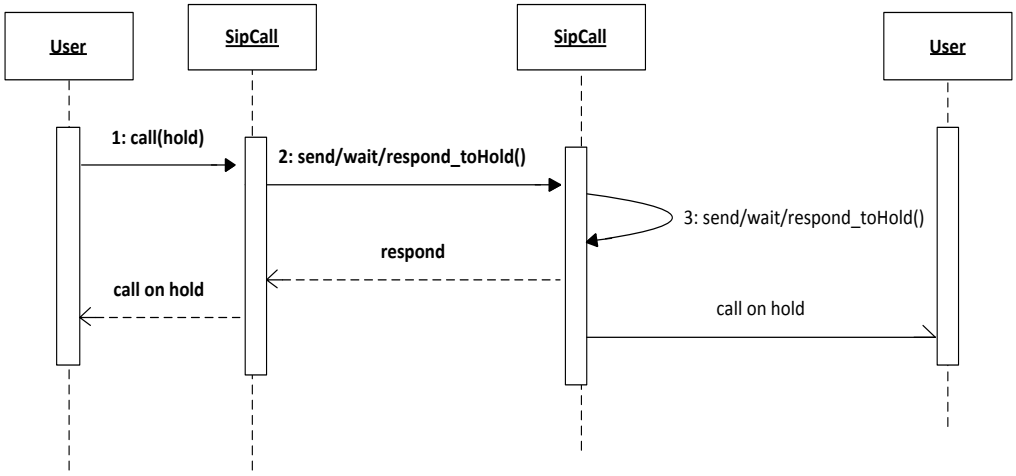
- **SUCCESSFUL FLOW**



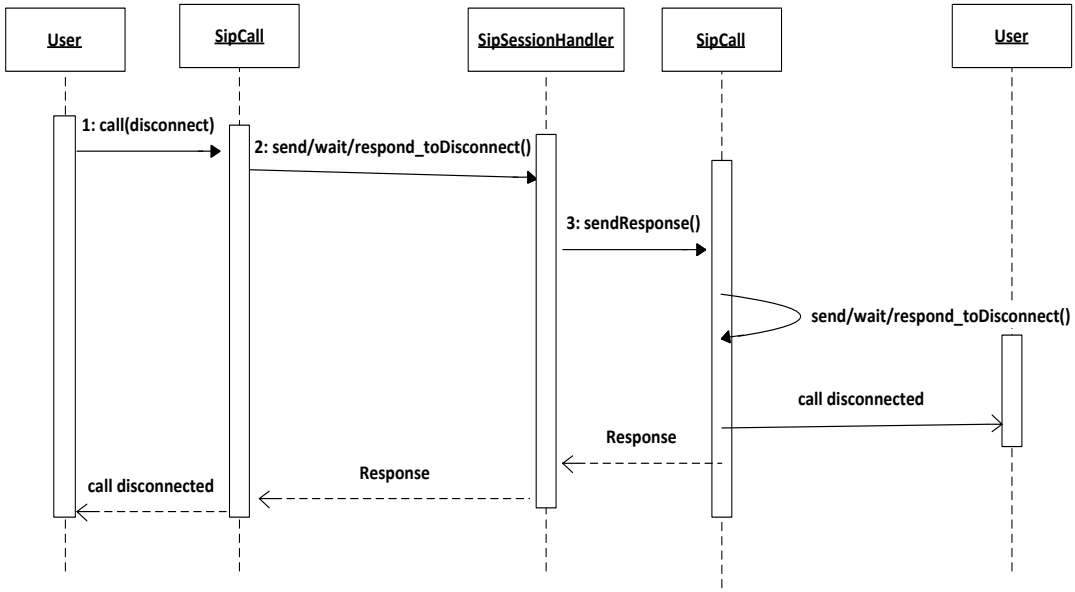
- **ALTERNATE FLOW**



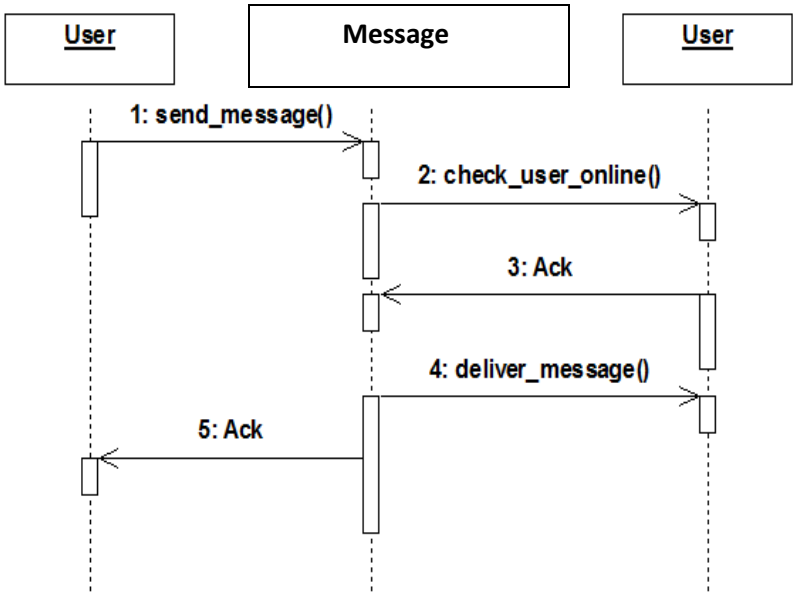
- **Hold Call**



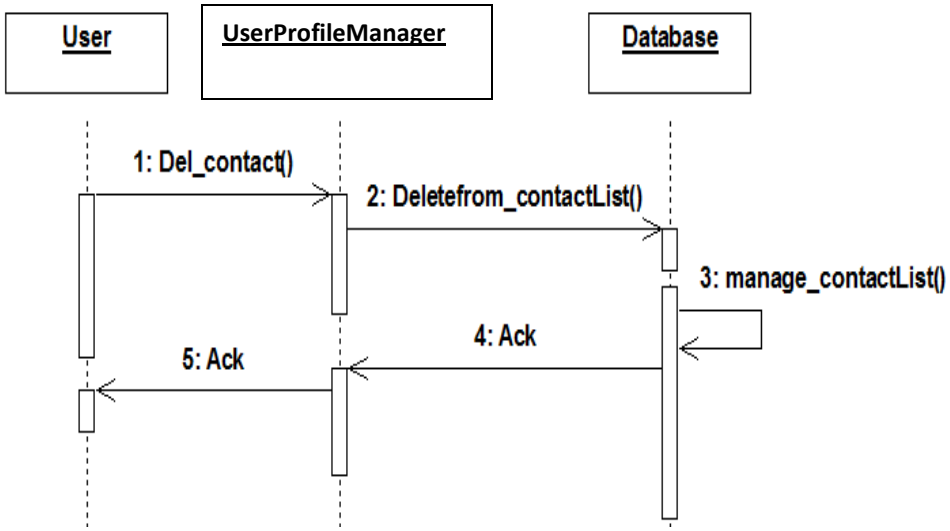
- **Disconnect Call**



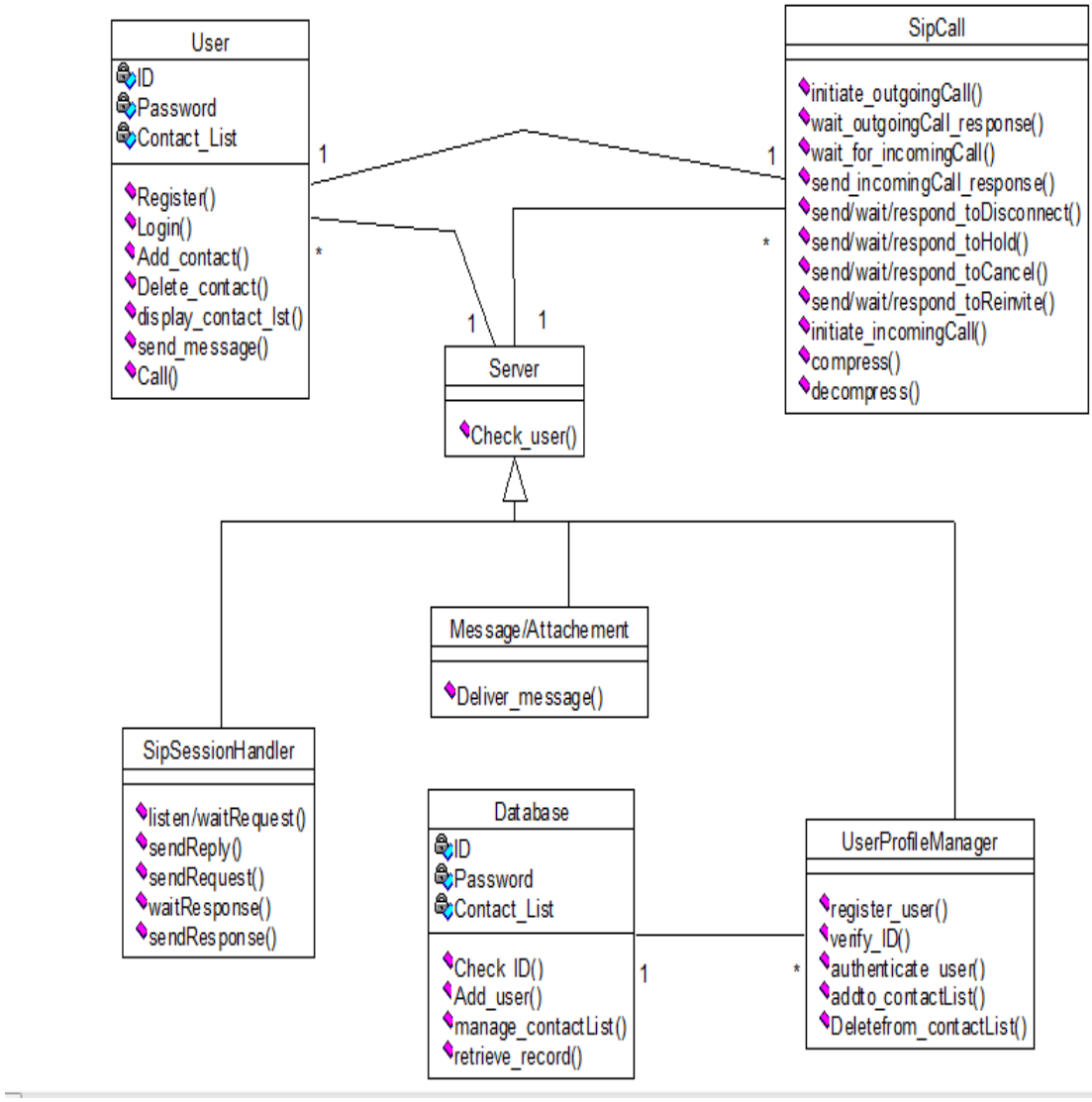
- **Send Message**



- **Delete Contact**



3.5) Class Diagram:-



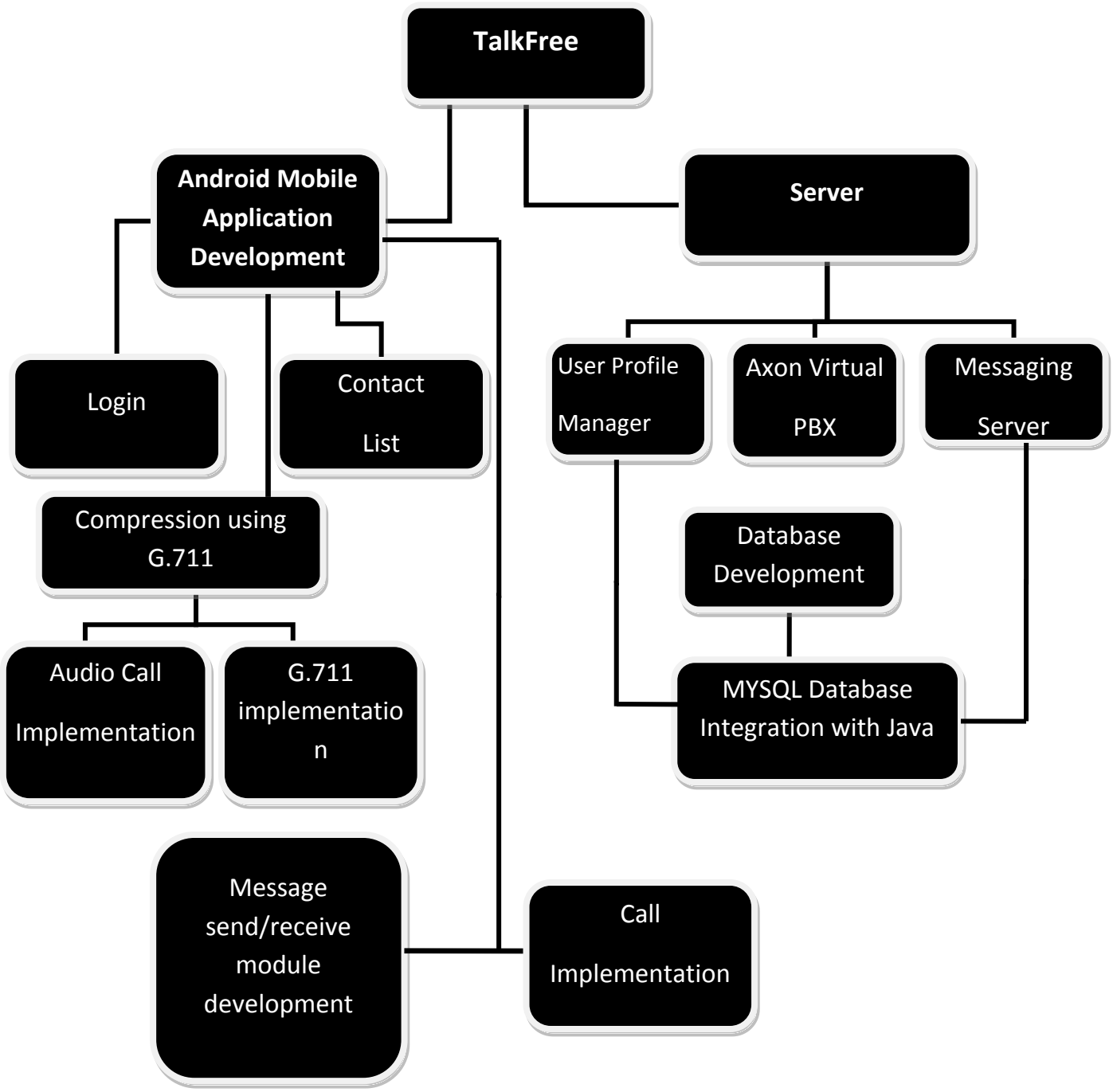
CHAPTER # 05: IMPLEMENTATION

5.1 Introduction

Implementation details of the project have been discussed in this chapter. The coding for the Messaging Server Application has been done in the Java in Netbeans 7.1.While for Android mobile client application coding is done in Java in Eclipse . As illustrated in the prior chapter, the major applications of the system are android mobile client and server which are divided into different Modules for flexible development and integration of inter-dependant Modules. The implementation details of the project are mentioned in next section.

5.2 Implementation

System implementation has been highlighted in the Figure 5-1 showing the overall architecture and design implementation of **TalkFree**.



5.3 Implementation Language

The implementation language which has been chosen for the Server and Android Mobile Client Application is java. It has been preferred over other languages because:

1. Java is an object oriented language.
2. It is compiled to an intermediate language (CIL) independently of the Language it was developed or the target machine architecture and operating System
3. Automatic garbage collection
4. Pointers no longer needed
5. Definition of classes and functions can be done in any order
6. Declaration of functions and classes not needed
7. Classes can be defined within classes
8. There are no global functions or variables, everything belongs to a class
9. All the variables are initialized to their default values before being used
10. (This is automatic by default but can be done manually using static constructors)

5.4 Distribution of Modules

5.4.1 Server Modules

The main Modules of the server application are:

5.4.1.1 User Profile Manager Module

This module is listening the connection from android mobile clients and responsible for registering and logging in users.This module also stores contact list of the user.

5.4.1.2 SIP Session Handler Module

This module is responsible for initiating session between user It works under the mechanism of SIP protocol

5.4.1.3 Messaging Module

This module is responsible for receiving messages from client application and stores them. Message is then delivered to other user by checking its online status.

5.4.2 Android Mobile Client Modules

5.4.2.1 Messaging Module

This module provides the functionality of sending message to other user. It allows user to type a text message and then send to other user. After sending message successfully it shows delivery status. It allows provide the functionality of inbox that shows all the received messages.

5.4.2.2 Compression Module

This module compresses the outgoing voice packets and decompress the incoming voice packets.

5.4.2.3 Call Module

This module is responsible establishing call with other user. It also provides the functionality of holding/unholding and disconnecting call.

5.5 Distribution of Classes with respect to Modules

5.5.1 Server Classes

TalkFree server application consists of following class.

5.5.1.1 MyServer Class

This class implements functionality of establishing connection with client and sending results of the query asked by the client. The main class is java class in which in the background keeps listening for TCP request to the server. Once a TCP connection is established it creates a new thread which manages all communication with that android mobile client and goes back to listening for new requests.

```
public static void main(String[] args) {...}
public static void view_con(String u_id) {...}
public static void check_inbox(String sender_id) {...}
public static String check_recv_id(String recv_id) {...}
public static void db_work(String recv_id, String mesg, String Sender_id) {...}
public static void set_status(String id) {...}
public static boolean login(String id, String pass) {...}
public static String check_id(String id) {...}
public static String check_friend(String id, String req_id) {...}
public static void set_table(String u_id, String c_id, String c_name) {...}
```

5.5.1.2 Authentication Class

This class verifies that username and password are correct and logins the user.

```
String QueryString1 = "select count(*) from information_schema.tables where table_schema='user_profile' and table_name='user'"; /
rs = statement.executeQuery(QueryString1);
 while (rs.next()) {
    // System.out.println("inside while");
    check = rs.getInt(1); //returns 0 if table does not exits and 1 if table exists in database

}
System.out.println(check);
if (check == 1) {

    QueryString2 = "SELECT * from user";
    rs = statement.executeQuery(QueryString2);
    while (rs.next()) {
        str = rs.getString(2);
        if (str.equals(id)) {
            check_pass = rs.getString(4);

        }

    }
    System.out.println(check_pass);
    System.out.println(pass);
    if (pass.equals(check_pass)) {
        resul = true;
    }
} else if (check == 0) {
    resul = false;
```

5.5.1.3 Contact List Class

This class retrieves contact list of the user from database and sends to user.

```
public static void view_contacts(String u_id) {
    try {

        String connectionURL = "jdbc:mysql://localhost:3306/user_profile";
        Connection connection = null;
        Statement statement = null;
        String str, QueryString2;
        ResultSet rs = null;
        int check = 2;
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
        connection = DriverManager.getConnection(connectionURL, "root", "12345");
        statement = connection.createStatement();
        String QueryString1 = "select count(*) from information_schema.tables where table_schema='user_profile' and table_name='" + u_id + "'";

        rs = statement.executeQuery(QueryString1);
        while (rs.next()) {
            check = rs.getInt(1); //returns 0 if table does not exits and 1 if table exists in database

        }

        if (check == 1) {

            // QueryString2 = "SELECT u_id , u_name from user where u_id='" + sender_id + "'";
            QueryString2 = "SELECT  * from " + u_id + "";

            System.out.println(QueryString2);
            rs = statement.executeQuery(QueryString2);
```

5.5.1.4 *Inbox Class*

This class provides the functionality of retrieving stored messages from database and sending messages to user.

```
            if (check == 1) {

                // QueryString2 = "SELECT u_id , u_name from user where u_id='" + sender_id + "'";
                QueryString2 = "SELECT  sender_name,messg from messages where recv_name='" + sender_id + "'";
                System.out.println(QueryString2);
                rs = statement.executeQuery(QueryString2);
                while (rs.next()) {
                    arry[i] = rs.getString(1);
                    i++;
                    arry[i] = rs.getString(2);
                    i++;

                }


            }

            rs.close();
            statement.close();
            connection.close();

        } catch (Exception ex) {
            System.out.println("Unable to connect to database.");
        }

    }
```

5.5.1.5 *AddContact Class*

This class checks that contact ID exists in the database and adds that contact successfully to the

contact list.

5.5.1.6 SendMessage Class

This message adds messages in the table of the received ID sent by other client.

```
if (check == 0) {
    QueryString1 = "CREATE TABLE messages(User_Id INTEGER NOT " + "NULL AUTO_INCREMENT, recv_name VARCHAR(25) " + ",sender_name VARCHAR(25) "
    updateQuery = statement.executeUpdate(QueryString1);

    QueryString2 = "INSERT INTO messages(recv_name,sender_name,messg) VALUES('" + recv_id + "','" + Sender_id + "','" + mesg + "' )";
    updateQuery = statement.executeUpdate(QueryString2);
    if (updateQuery != 0) {
        System.out.println("New table is created successfully and message is added.");
    }

} else if (check == 1) {


    QueryString1 = "INSERT INTO messages(recv_name,sender_name,messg) VALUES('" + recv_id + "','" + Sender_id + "','" + mesg + "' )";
    updateQuery = statement.executeUpdate(QueryString1);
    if (updateQuery != 0) {
        System.out.println("Message successfully is added.");
    }

}


QueryString2 = "SELECT * from messages";
rs = statement.executeQuery(QueryString2);
while (rs.next()) {
    System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3) + "  " + rs.getString(4) + "\n");
}
```

.

5.5.2 Android Mobile Client Classes

TalkFree Mobile application comprises of following classes

5.5.2.1 Login Class

This class establishes connection with server and sends the username and password entered by the user to server for authentication. On successful authentication it takes user to home screen.

```
    u_id=(EditText) findViewById(R.id.u_id);
    passw=(EditText) findViewById(R.id.passw);
    domain=(EditText) findViewById(R.id.domain);

    next.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            //domain1=domain.getText().toString();
         Socket socket = null;
DataOutputStream dataOutputStream = null;
DataInputStream dataInputStream = null;

try {

socket = new Socket(domain.getText().toString(), 8888);
dataOutputStream = new DataOutputStream(socket.getOutputStream());
dataInputStream = new DataInputStream(socket.getInputStream());
dataOutputStream.writeUTF("login");
dataOutputStream.writeUTF(u_id.getText().toString());
dataOutputStream.writeUTF(passw.getText().toString());
text=dataInputStream.readUTF();

} catch (UnknownHostException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
} catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}
```

5.5.2.2 Send Message Class

This class provides the functionality of allowing user to type message and then sends this message to server with receiver id.

```
 stringRecd = intent.getStringExtra("name");
 server_domain = intent.getStringExtra("server");
 next.setOnClickListener(new View.OnClickListener() {
     public void onClick(View view) {

      Socket socket = null;
      DataOutputStream dataOutputStream = null;
      DataInputStream dataInputStream = null;

      try {
       socket = new Socket(server_domain, 8888);
       dataOutputStream = new DataOutputStream(socket.getOutputStream());
       dataInputStream = new DataInputStream(socket.getInputStream());
      dataOutputStream.writeUTF("send");
      dataOutputStream.writeUTF(textOut.getText().toString());
      dataOutputStream.writeUTF(recv_id.getText().toString());
      dataOutputStream.writeUTF(stringRecd);
       // dialog has to b shown
       frmServer=dataInputStream.readUTF();
  //      Toast.makeText(Send.this, frmServer, Toast.LENGTH_LONG).show();
      } catch (UnknownHostException e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      } catch (IOException e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      }
```

5.5.2.3 Check Inbox Class

This class provides the functionality of retrieving messages from server and displaying to user.

```
    Button next1 = (Button) findViewById(R.id.button1);
    Intent intent = getIntent();

    /**retrieve the string extra passed*/
    stringRecd = intent.getStringExtra("name"); // login name
    server_domain = intent.getStringExtra("server"); // server
        buttonSend.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
Socket socket = null;
DataOutputStream dataOutputStream = null;
DataInputStream dataInputStream = null;
    try {

        socket = new Socket(server_domain, 8888);
        dataOutputStream = new DataOutputStream(socket.getOutputStream());
        dataInputStream = new DataInputStream(socket.getInputStream());
        dataOutputStream.writeUTF("inbox");
        dataOutputStream.writeUTF(stringRecd);
        sender1.setText(dataInputStream.readUTF());
        mesg1.setText(dataInputStream.readUTF());
        sender2.setText(dataInputStream.readUTF());
        mesg2.setText(dataInputStream.readUTF());
        sender3.setText(dataInputStream.readUTF());
        mesg3.setText(dataInputStream.readUTF());
```

5.5.2.4 Add Contact Class

This class provides the functionality of adding contact to contact list. This class sends the contact name and id entered by the user to server and displays the result on verification of contact.

```
public static String check_friend(String id, String req_id) {
    String u_name = null;
    try {

        String connectionURL = "jdbc:mysql://localhost:3306/user_profile";
        Connection connection = null;
        Statement statement = null;
        String str, QueryString2;
        ResultSet rs = null;
        int check = 2;
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
        connection = DriverManager.getConnection(connectionURL, "root", "12345");
        statement = connection.createStatement();
        String QueryString1 = "select count(*) from information_schema.tables where table_schema='user_profile' and table_name='" + id + "'";

        rs = statement.executeQuery(QueryString1);
        while (rs.next()) {
            check = rs.getInt(1); //returns 0 if table does not exits and 1 if table exists in database

        }
        if (check == 1) {

            QueryString2 = "SELECT * from " + id + "";
            rs = statement.executeQuery(QueryString2);
            while (rs.next()) {
                str = rs.getString(2);
                if (str.equals(req_id)) {
                    u_name = "Contact already exist";
                }
```

5.5.2.5 G711 Class

The G711 class implements the G711 algorithm for call compression .This class provides methods for U-Law , A-Law and linear PCM.

```
//change G711 ulaw start
static final int _u2a[] = {                                  /* u- to A-law conversions */
    1,      1,      2,      2,      3,      3,      4,      4,
    5,      5,      6,      6,      7,      7,      8,      8,
    9,      10,     11,     12,     13,     14,     15,     16,
    17,     18,     19,     20,     21,     22,     23,     24,
    25,     27,     29,     31,     33,     34,     35,     36,
    37,     38,     39,     40,     41,     42,     43,     44,
    46,     48,     49,     50,     51,     52,     53,     54,
    55,     56,     57,     58,     59,     60,     61,     62,
    64,     65,     66,     67,     68,     69,     70,     71,
    72,     73,     74,     75,     76,     77,     78,     79,
    81,     82,     83,     84,     85,     86,     87,     88,
    89,     90,     91,     92,     93,     94,     95,     96,
    97,     98,     99,     100,    101,    102,    103,    104,
    105,    106,    107,    108,    109,    110,    111,    112,
    113,    114,    115,    116,    117,    118,    119,    120,
    121,    122,    123,    124,    125,    126,    127,    128};

static final int _a2u[] = {                                  /* A- to u-law conversions */
    1,      3,      5,      7,      9,      11,     13,     15,
    16,     17,     18,     19,     20,     21,     22,     23,
    24,     25,     26,     27,     28,     29,     30,     31,
    32,     32,     33,     33,     34,     34,     35,     35,
    36,     37,     38,     39,     40,     41,     42,     43,
    44,     45,     46,     47,     48,     48,     49,     49,
    50,     51,     52,     53,     54,     55,     56,     57,
```

5.5.2.6 Call Class

The Call class implements SIP calls. It handles both outgoing and incoming calls. This class provides the functionality of listening for incoming call , rings on incoming call , responses to incoming call and rejecting incoming call. This class also implements the functionality of inviting other user for outgoing call and waiting for response and establishing outgoing call.

```
        dialog.inviteWithoutOffer(callee, from, contact);
    }

    /** Starts a new call with the <i>invite</i> message request */
    public void call(Message invite) {

    /** Answers (in the ack message) */
    public void ackWithAnswer(String sdp) {

    /** Rings back for the incoming call */
    public void ring(String sdp) { // modified

    /** Respond to a incoming call (invite) with <i>resp</i> */
    public void respond(Message resp) {

    /** Accepts the incoming call */
    /*
     * public void accept() { accept(local_sdp); }
     */

    /** Accepts the incoming call */
    public void accept(String sdp) {
```

5.5.2.7 CallListner Class

This class is implemented to manage sip calls. Objects of Call class uses methods of this class to signal specific call events.

```
        /** Callback function called when arriving a 180 Ringing */
        public void onCallRinging(Call call, Message resp);

        /** Callback function called when arriving a 2xx (call accepted) */
        public void onCallAccepted(Call call, String sdp, Message resp);

        /** Callback function called when arriving a 4xx (call failure) */
        public void onCallRefused(Call call, String reason, Message resp);
```

5.5.2.8 CallListnerAdapter Class

Class CallListenerAdapter implements CallListener interface providing a dummy implementation of all Call callback functions used to capture Call events.

```
 */
public void onCallIncoming(Call call, NameAddress callee,
        NameAddress caller, String sdp, Message invite) { // printLog("INCOMING");
    String local_session;
    if (sdp != null && sdp.length() > 0) {
        SessionDescriptor remote_sdp = new SessionDescriptor(sdp);
        SessionDescriptor local_sdp = new SessionDescriptor(call
                .getLocalSessionDescriptor());
        SessionDescriptor new_sdp = new SessionDescriptor(remote_sdp
                .getOrigin(), remote_sdp.getSessionName(), local_sdp
                .getConnection(), local_sdp.getTime());
        new_sdp.addMediaDescriptors(local_sdp.getMediaDescriptors());
        new_sdp = SdpTools.sdpMediaProduct(new_sdp, remote_sdp
                .getMediaDescriptors());
        new_sdp = SdpTools.sdpAttirbuteSelection(new_sdp, "rtpmap");
        local_session = new_sdp.toString();
    } else
        local_session = call.getLocalSessionDescriptor();
    call.ring(local_session);
    // accept immediatly
    call.accept(local_session);
}

/**
```

5.6 Deliverables:

1. TalkFree Server Application
2. TalkFree Android Mobile Client Application

5.7 Summary

Details of the classes implemented have been discussed in this chapter. The classes have been distributed among the two basic components of the system which are server and android mobile client. Java has been used as the programming language for the project due to its object-oriented and platform independent nature.

CHAPTER # 06: TESTING

Project testing has been divided into unit testing and system integration testing.

6.1 UNIT TESTING

The system has following Unit Test Cases:

REQUIREMENT NO 1) GUI

- **Test data entered in ID textbox**

PURPOSE: To test user has entered value in text box

PRE-REQUISITES: System is running

TEST DATA: {valid user ID, numbers, Any characters}

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Predefined ID	No error message	No error message	Pass
2.	Any Characters	Enter numbers only	Enter numbers only	Pass

STEPS

Screen showing text box to enter ID

User will enter his/her ID

- **Test data entered in Password textbox**

PURPOSE: To test user has entered value in text box

PRE-REQUISITES: System is running

TEST DATA: {valid user Password, numbers, Any characters}

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Predefined ID	No error message	No error message	Pass
2.	Any Characters or numbers	No error message	No error message	Pass

STEPS

Screen showing text box to enter password

User will enter his/her password

- **Test data entered in Domain text box**

PURPOSE: To check user has entered correct data

PRE-REQUISITES: System is running

TEST DATA: { numbers, empty}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	digits	No error message	No error message	Pass
2.	Any Characters	Enter numbers only	Enter numbers only	Pass
3.	empty	Invalid domain	Invalid domain	Pass

STEPS

Screen showing text box to enter domain

User will enter his/her domain

- **Test functionality Login Button**

PURPOSE: if functionality and entered details are then user will able to login.

PRE-REQUISITES: system is running and details are entered.

TEST DATA: {click on button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	button is pressed	Home screen is displayed	Home screen is displayed	Pass

STEPS

Screen showing text boxes and login button

User will enter details

Login button is pressed

- **Test functionality of Login button (in case for login is not successful)**

PURPOSE: To verify checks for in valid login

PRE-REQUISITES: System is running, and user has entered his /her incorrect information.

TEST DATA: {click on Login button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Login button is pressed	Error is displayed "Invalid Login"	Error message is displayed	Pass

STEPS

User entered his/her ID , Password or domain incorrectly

Press Login button

“Invalid Login” ,message is displayed

Text boxes are cleared

User is suppose to entered his/her information again

****

Home Screen:

- **Test Send Message Button**

  PURPOSE: To check the proper functionality of this button

  PRE-REQUISITES: System is running, login is successful

  TEST DATA: Click on Send Message button

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Button is pressed	Activity is changed	Activity is changed	Pass

STEPS

Login is Successful

User will press the Send Message button

- **Test Inbox button**

  PURPOSE: To test user is able see receive messages.

  PRE-REQUISITES: System is running and login is successful

  TEST DATA= {click on Inbox button on screen}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Button is pressed	Activity is changed	Activity is changed	Pass

STEPS:

Login is Successful

User will press the Inbox button

- **Test Add Contact Button**

PURPOSE: To test user is able to add a new contact.

PRE-REQUISITES: System is running and login is successful

TEST DATA= {click on Add Contact button on screen}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Button is pressed	Activity is changed	Activity is changed	Pass

STEPS:

Login is Successful

User will press the Add Contact button

- **Test Contact List Button**

PURPOSE: To test user is able see his/her contact list.

PRE-REQUISITES: System is running and login is successful

TEST DATA= {click on Contact List button on screen}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Button is pressed	Activity is changed	Activity is changed	Pass

STEPS:

Login is Successful

User will press the Contact List button

****

Send Message:

- **Test Data entered To text box**

  PURPOSE: To test user has entered value in text box is according to specified format.

  PRE-REQUISITES: System is running, successful login, user select send message option.

  TEST DATA= {any amount in digits, any character, empty}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Digits	Show error "Characters only"	Show error "Characters only"	Pass
2.	Any character	No error message	No error message	Pass
3.	Empty	Error "Invalid recipient"	Error "Invalid recipient"	Pass

  STEPS:

  Screen showing text box to enter receiver name

  User entered the name

- **Test Data entered "Text" text box**

  PURPOSE: To test user has entered value in text box is according to specified format.

  PRE-REQUISITES: System is running, successful login, user select send message option.

  TEST DATA= {any amount in digits, any character, empty}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Digits	No error message	No error message	Pass

2.	Any character	No error message	No error message	Pass
3.	Empty	Error “Invalid Message”	Error “Invalid Message”	Pass

STEPS:

Screen showing text box to enter message

User entered the message

- **Test functionality of Send button (in case of correct info)**

  PURPOSE: if functionality is correct message is successfully added to database,

  PRE-REQUISITES: System is running and login is successful.

  TEST DATA: {click on Send button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Send button is pressed	“Message Sent ” this message is shown	“Message Sent ” this message is shown	Pass

STEPS

Successful login

User select send message option

User enter receiver name and desire message

Send button is pressed

- **Test functionality of Enter button (in case of Incorrect info)**

  PURPOSE: to check what if user entered incorrect receiver name or message.

  PRE-REQUISITES: System is running and login is successful, receiver name and message is entered.

  TEST DATA: {click on Send button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Send button is pressed	One of the following messages can appear "Invalid Recipient", "Invalid Message".	One of the following messages can appear "Invalid Recipient", "Invalid Message".	Pass

STEPS

Successful login

User select send message option

Info is added

User press Send button

****

Inbox:

- **Test functionality of Show button**

  PURPOSE: if functionality is correct messages are displayed to user.

  PRE-REQUISITES: System is running and login is successful.

  TEST DATA: {click on Show button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Show button is pressed	Display receive messages	Display receive messages	Pass

STEPS

Successful login

User select Inbox option

User press Show button

- **Test functionality of Home button**

  PURPOSE: to check what if user press Home button.

  PRE-REQUISITES: System is running and login is successful,

  TEST DATA: {click on Home button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Home button is pressed	Display Home Screen	Display Home Screen	Pass

STEPS

Successful login

User press Home button

****

Add Contact:

- **Test data entered in Contact ID textbox**

  PURPOSE: To test user has entered value in text box

  PRE-REQUISITES: System is running

  TEST DATA: {valid user ID, numbers, Any characters}

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Predefined ID	No error message	No error message	Pass
2.	Any Characters	Enter numbers only	Enter numbers only	Pass

STEPS

Screen showing text box to enter ID

User will enter his/her ID

- **Test data entered in Contact Name textbox**

PURPOSE: To test user has entered value in text box

PRE-REQUISITES: System is running

TEST DATA: {Any characters, numbers}

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Numbers	"Enter alphabets only"	"Enter alphabets only"	Pass
2.	Any Characters	No error message	No error message	Pass

STEPS

Screen showing text box to enter name

User will enter contact name

- **Test functionality Add Button**

PURPOSE: if functionality and entered details are correct then contact will be added.

PRE-REQUISITES: system is running and details are entered.

TEST DATA: {click on Add button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	button is pressed	Home screen is displayed	Home screen is displayed	Pass

STEPS

Screen showing text boxes and login button

User will enter details

Add button is pressed

****

Contact List:

- **Test functionality Contact List Button**

  PURPOSE: if functionality is correct then contact list will be displayed.

  PRE-REQUISITES: system is running.

  TEST DATA: {click on Contact List button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	button is pressed	Contact list is displayed	Contact list is displayed	Pass

STEPS

Contact List button is pressed

****

Call:

- **Test functionality Call Button**

  PURPOSE: if functionality is correct then call is establish.

  PRE-REQUISITES: system is running.

  TEST DATA: {click on Contact List button}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Call button is pressed	Activity changed	Activity changed	Pass

STEPS

Call

****

REQUIREMENT) LOGIN

- **Test User ID entered by the user**

PURPOSE: To test user has entered correct user ID or not.

PRE-REQUISITES: System is running

TEST DATA= {valid user name, any character}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Correct input (ID)	Login Successful	Login Successful	Pass
2.	Any Characters (a, b, c…)	Enter numbers only	Enter numbers only	Pass

STEPS:

Screen showing text box to enter user ID

Customer will enter his/her ID

- **Test Password entered by the user**

PURPOSE: To check user has entered correct password or not

PRE-REQUISITES: System is running and

TEST DATA: {valid user name, any character, and numbers}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL

1.	Correct Password	Login Successful	Login Successful	Pass

STEPS

Screen showing text box to enter password

User will enter his/her password

- **Successfully Login User**

  PURPOSE: After entering correct ID and password, user should be logged in

  PRE-REQUISITES: User ID and password entered by the user

  TEST DATA: {valid user name, any character, and numbers, empty}

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Correct details	Login Successful	Login Successful	Pass
2.	Incorrect details	"Invalid Login"	"Invalid Login"	Pass

STEPS

User ID and password entered by the customer

Match user ID and password with records in database

Match successful , login user

****

REQUIREMENT) Call

- **Test callee ID entered by the User**

PURPOSE: To test user has entered correct contact ID or not for call.

PRE-REQUISITES: System is running and callee is online

TEST DATA= {valid user ID, any character}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Correct input (ID)	Call established	Call established	Pass
2.	Any Characters (a, b, c…)	Enter numbers only	Enter numbers only	Pass

STEPS:

Screen showing text box to enter callee ID

user will enter callee ID

- **Test Call Establishment**

  PURPOSE: After entering correct ID and if the person having this id is online then call should be establish.

  PRE-REQUISITES: User ID entered by the user and that person is online

  TEST DATA: {valid user name, any character, and numbers, empty}

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Valid ID and Online user	Call Established	Call Established	Pass
2.	Valid ID and offline user	Error “User is not available”	Error “User is not available”	Pass

STEPS

User ID and password entered by the customer

Match user ID and password with records in database

Match successful , login user

****

- **Test Hold Call**

  PURPOSE: To check Hold Call button functionality

  PRE-REQUISITES: User is login and call is established

  TEST DATA: {Press Hold Call button}

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Hold Call button is pressed	Call on hold	Call on hold	Pass

STEPS

User press Hold button

****

- **Test Disconnect Call**

  PURPOSE: To check Disconnect Call button functionality

  PRE-REQUISITES: User is login and call is established

  TEST DATA: {Press Hold Call button}

TEST CASE ID	INPUT	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Disconnect Call button is pressed	Call is disconnected	Call is disconnected	Pass

STEPS

User press Disconnect Call button

REQUIREMENT) Send Message

- **Check whether receiver name entered correct or not.**

PURPOSE: To check user has entered the right receiver name

PRE-REQUISITES: System is running

TEST DATA: {valid user name, any character, and numbers, empty}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Enter name	Successful	Successful	Pass
2.	Enter numbers	Invalid ID	Invalid ID	Pass

STEPS

User will select Send Message

Send Message Screen will appear

The user will enter the receiver name

Entered name is checked against records in database.

User is notify accordingly

****

REQUIREMENT NO) Add Contact

- **To check that the contact is added or not.**

PURPOSE: To check that new contact is added or not.

PRE-REQUISITES:

System is running

User is successfully logged in

TEST DATA: {valid contact ID, any character, and numbers, empty}

TEST CASE ID	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/ FAIL
1.	Enter valid contact ID	Contact is added into database and list is updated	Contact is added into database and list is updated	Pass
2.	Enter Invalid ID	Error "Invalid Contact "	Error "Invalid Contact "	Pass
3.	Enter existing contact ID	Error "Contact already exist"	Error "Contact already exist"	Pass

STEPS

User will select Add Contact

Add Contact screen will appear

User enters contact name and ID

Entered ID is checked against the records in database to verify that the user exists or not

If Exists "Contact is successfully added to the list"

Or display a message "Invalid Contact"

This will also check for whether the entered contact is already exists in list or not.

If it is already in list then error will be displayed "Contact already exists".

Else Successfully "Contact is successfully added to the list"

2) INTEGRATION TESTING

- **Entry Criteria**
  All functions have been unit tested.

- **Elements to be integrated**
  User Interface, Login verification, Call, Send Message, Inbox, Add Contact ,Contact List
- **Integration Strategy**
  Big Bang strategy is used to integrate the modules.

- **Software Integration Sequence**
  Given below is the integration sequence.

- **Module Integration in testing phases:**

  - ✓ **First Phase**

    In this phase modules Login, Call, Send Message, Inbox, Add Contact, and Contact List are individually tested for the functionality they provide.

  - ✓ **Second Phase**

    In this phase Login module is integrated through Interface module to provide combined functionality.

  - ✓ **Third Phase**

    At this phase along with Login module Call, Send Message, Inbox, Add Contact and Contact List modules are added to the system to provide combined functionality.

CHAPTER 7 :USER MANUAL – TalkFree

7.1 System Requirements:

a. Windows XP for Server

b. Android Mobile Phone with minimum gingerbread 2.3

7.2 Installation:

The Installation of the TalkFree is Simple. There are two type of TalkFree Module

7.2.1 Server Installation:

a. Put the folder containing the Software in the Root Directory of C Drive of the Computer.

b. Name the root folder containing all the Software files " TalkFree"

c. Install the server

d. Change firewall settings and allow incoming and outgoing connections on port 5061.

7.2.2 Client Installation:

e. Transfer talkfree.apk file to mobile phone and click the file for installation.

7.3 How to use the software?

a. Server Module:

a. **Running The Server:**

i. Run the server.exe

ii. In Login screen, Enter the username and password (By default Username: admin Password: 123123).

iii. For changing settings of the calls, click on web panel and change configurations accordingly.

iv. Press Ok to save setting.

b. **Stopping Server:**

i. Press The "Stop Server" Button"

c. **Allowing User to Connect the Server:**

i. Press the button "Run Server".

ii. In status box, a message appears "server started".

iii. Now if any clients try to connect to the server. A message will be shown in the status box.

iv. If any client establishes calls other user status box shows the status

v. When call is established and ended status box shows the status accordingly.

d. **User Management:**

i. Click the "Add Extensions "Button at web panel.

ii. **Add Extension:** For adding new user, click the "Add extension" button. Enter the desired username, password and other configuration fields.

iii. **Delete User:** For deleting the existing user press the delete button and confirm deletion

iv. **Reset Password:** You can also reset the user password by using reset button.

b. Client Side:

a. Login:

i. Enter user name, password and server ip address and press login.

ii. On successful login, you will be taken to home screen that will display contact list, add contact and inbox.

iii. If Error message appears then check your server IP or Status of the server from network admin.

b. Send Message:

i. Press the message button in front of the user name in contact list

ii. Type the message and press send button

iii. On successful message delivery , application will display notification "message delivered successfully"

iv. If message is not delivered, application will display notification accordingly.

c. Call User

i. Press the Call button in front of user in contact list

ii. Application will display outgoing call status

iii. On successful receiving of call , application will display call in progess

d. Add Contact

i. Click on Add Contact button on home screen

ii. Enter name and ID and press add contact button

iii. On successful addition of contact in the list , updated contact list will be displayed

CHAPTER # 08:

- **CONCLUSION**

Today mobile VoIP calling is a major concern in most of the public and private sector organizations where cost of calling through cellular networks really matters. TalkFree is designed to facilitate the audio calling via internet. It also provides the functionality of sending and receiving messages to other users. Further more for enhancing voice quality it uses G711 compression techniques. Application also enables user to add contact so user can easily send message or call in case if user forgets the ID of the user.

- **FURTUE WORK**

Mobile VoIP deployments are going to be challenging for service providers in the future. Service providers need to embrace the needs and requirements of all of the different mobile carrier networks as well as software development for all of the different devices. Following enhancements can be made

- Develop and maintaining software to integrate and work well for all type of mobile operating systems
- New algorithms can be developed that can compress voice and provide best voice quality even on congested or worse networks
- Develop sip server
- Extend the functionality of messages by allowing users to attach files of different formats
- Extend the functionality of calls to conference calls

REFERENCES

http://www.voip-info.org/wiki/view/SIP

http://www.siptutorial.net/SIP/

http://developer.android.com/resources/tutorials

http://www.voip-info.org/wiki/view/ITU+G.711

http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml

http://www.javvin.com/protocolG7xx.html

http://www.freesoft.org/CIE/Topics/127.htm

http://www.skill-guru.com/blog/2011/01/10/getting-started-with-android-development/

http://www.learnerstv.com/Free-Computers-Video-lectures-ltv461-Page1.htm

http://www.javaworld.com/javaworld/jw-06-2006/jw-0619-sip.html

http://java.sun.com/products/jain/SIP-and-Java.html

http://jsip.java.net/

http://p2p-sip.blogspot.com/2009/01/programming-languages-for-implementing.html

http://www.daniweb.com/software-development/java/threads/28037/how-to-implement-sip-in-java-platform