**Chapter 1**

**INTRODUCTION**

**1.1 Introduction**

Smart phones are now very common devices to have. Nowadays Smart phones are replacing the tradition mobile phones. Smart phone controlling is one of the leading concepts of android application. Sometimes we need to access the smart phone remotely and get the contents of the phone. The information should be sent with the least cost and efficiently.

Smart phones are costlier than the ordinary phones. We sometimes leave the phone on table of a restaurant by hurry or negligence, so need an assistant or something like that to alert us to take the phone.

So to achieve the above two concept the “An Intelligent Smart Phone Controller” which will be an android application will be helpful. It will give and assistant support and also remote accessing to the user.

**1.2 Objective and Goals**

There are several remote accessing applications available in the android market, but all those application needs an active internet facility. Those applications also need a smart phone to access another smart phone. And they have their own disadvantages.

So the objective is to develop an android application which helps to access the smart phone through a different non smart phone, and also to use simple SMS service available in the mobile. Another objective is that application must be installed only in smart phone and no need to install anything in the remote phone.

**1.3 Proposed System**

To overcome the above mentioned disadvantage here is an alternate proposal for solving the above problem through an android application which does not need an active internet connection. Here the information is accessed through the normal Messages and the application is activated through the message. Before allowing the remote user to access the phone, the application does some authentication process.

In addition to the remote accessing we have a mobile remainder utility through which we can minimize the rate of losing the smart phone. Here we are going to use a Bluetooth device which will be in continuous contact with the paired smart phone, if the connection is lost then the Bluetooth device starts to vibrate. Again if connection is gained then the vibration stops indicating that the smart phone is near the user.

**1.4 Features**

Some of the features of the proposed android application are,

* Accessing the contacts of the smart phone from the remote phone through messages.
* Accessing the missed calls of the smart phone from the remote phone through messages.
* Accessing the messages of the smart phone from the remote phone through messages.
* Remotely ringing the smart phone when it is silent mode through simple message.
* Changing the profile of the smart phone through simple messages.
* Getting the location of the smart phone remotely through message.
* Connecting and pairing with the external Bluetooth device for Bluetooth reminder.

**Chapter 2**

**SOFTWARE REQUIREMENTS SPECIFICATION**

**2.1 Introduction**

**2.1.1 Purpose**

The purpose of software requirement specification of the system is to give a feasible solution to the organization and it is the main focus of this document. The emphasis in requirement analysis is identifying what is needed from system.

The “Intelligent Smart Phone Controller” is an application used to control the smart phone remotely; it also alerts the owner of the smart phone not to leave the phone unattended i.e. it helps to reduce the smart phone missing or theft.

**2.1.2 Scope**

We describe that what features are in the scope of the software and what are not in the scope of the software to be developed.

In scope

* Accessing the messages of one smart phone from another smart phone.
* Accessing the contact details of one smart phone from another smart phone.
* Accessing the call log of one smart phone from another smart phone.

Out of scope

* Calling a number remotely.
* Accessing the multimedia files from the remote Smart phone.

**2.1.3 Definitions, Acronyms, and Abbreviations**

Acronyms and Abbreviations

1. **ISPC**: Intelligent Smart Phone Controller
2. **SRS**: Software Requirements Specification.

Definitions

1. **Bluetooth**: It is a device which works on the wireless technology i.e. short wave length radio waves.
2. **Microcontroller**: It is a small computer on a single integrated circuit containing processor core, memory, and programmable input/output peripherals.
3. **Vibrator**: It is a motor fitted to the Bluetooth device to give the vibrating alert.
   * 1. **Overview**

The rest of this SRS is organized as follows: Section 2 gives an overall description of the software. It gives what level of proficiency is expected of the user, some general constraints while making the software and some assumptions and dependencies that are assumed. Section 3 gives specific requirements which the software is expected to deliver. Functional requirements are given by various use cases. Some performance requirements and design constraints are also given.

* 1. **Overall Description**

**2.2.1 Product Perspective:**

**ISPC** is aimed towards the controlling the smart phone and to help the user to access his smart phone remotely when needed. It helps the user to get the information from his phone when needed. It works on the short message service of the phone.

**2.2.1.1Product Functions:**

**ISPC** will have two utilities, i.e. one for Bluetooth reminder and the second is remote accessing. In the first utility, i.e. Bluetooth reminder an option for turning ON/OFF the functionality will be provided. After turning ON the option, the mobile’s Bluetooth will be turned ON and the mobile device will authenticate with the Bluetooth device. After authentication is successful, the Bluetooth reminder functionality will be activated. In the second utility, i.e. Remote accessing, a message will be sent by the other mobile to the smart phone for activating the remote accessing functionality. The application will do some sort of authentication and then allow the user to access his smart phone from another mobile remotely.

|  |  |  |
| --- | --- | --- |
| **Class of use cases** | **Use cases** | **Description of use cases** |
| Use case related to installation and sign up | Installation | Creates and Initializes working files |
| Sign up | Saves the user details like name password and email |
| Use case related to application authorization | Login | Login to **ISPC** |
| Forget Password | Password will be sent to mail |
| Use case related to remote mobile authentication | Connect | Creates an connection with smart phone |
| Disconnect | Removes the connection with smart phone |
| Use case related to remote accessing | Contact retrieval | Retrieves the contact detail from smart phone |
| Missed call retrieval | Retrieves the missed call from the smart phone |
| Message retrieval | Retrieves the unread message from the smart phone |
| Ringer | Rings the smart phone when in silent mode |
| Profile Changer | Changes the profile of the smart phone from silent to general and vice versa |
| Location tracker | Sends the location of the smart phone to the remote phone |
| Use case related to Bluetooth | Bluetooth remainder | Reminds the user not leave his smart phone unattended |

Table 2.1 Product Functions

* + 1. **User Characteristics:**

User must be familiar with the mobile functionalities.

1. User must have knowledge of how to send short message services.
2. User must have a Short message service in his mobile.
   * 1. **General Constraints:**
3. **ISPC** is a Single-User Software.
4. Phone must have android operating system.
5. Phone must have messaging facility.
6. Phone must be switched on i.e. it must not be switched off.
   * 1. **Assumptions and Dependencies:**
7. Working of the Bluetooth reminder depends on the correct Bluetooth functionality of the mobile and also of the Bluetooth device.
8. Working of the remote accessing functionality depends on the mobile network and also the battery life of the mobile.
   1. **Specific Requirements**
      1. **Functional Requirements:**

**ISPC** is an application which will be used by a single person, so the actor of the system will be the User or Owner of the smart phone. User of the mobile is only person who can access the application.

We describe Functional Requirements by giving various use cases

Use case related to Installation*:*

**Use case 1:** Installation

Primary Actor*:* User

Pre Condition: Nil

Main Scenario:

1. User initiates **ISPC** installation program.
2. **ISPC** asks for the acceptance of the services that will run on phone.
3. **ISPC** application will be installed on the smart phone.

**Use case 2:** Sign up

Primary Actor: User

Pre Condition: **ISPC** application must be correctly installed.

Main Scenario:

1. User opens the **ISPC** application.
2. **ISPC** asks for the Username, Password, and the email address of the user.
3. **ISPC** stores the details and opens the first screen.

Use case related to Application authentication:

**Use case 3:** Login

Primary Actor: User

Pre condition: **ISPC** application must be correctly installed.

Main Scenario:

1. User opens the **ISPC** application.
2. **ISPC** asks for the password.
3. User enters the password.
4. **ISPC** checks for correctness, record matches and the first screen is displayed.

Alternate Scenario*:*

1. Authentication fails.
2. User is requested to retype the password.

**Use case 4:** Forgot Password

Primary Actor: User

Pre-Condition: **ISPC** application must be correctly installed and internet facility must be present.

Main Scenario*:*

1. User initiates forgot password.
2. Password will be sent to the email address specified by the user.

Use case related to Remote mobile authentication:

**Use case 5:** Connection

Primary Actor: User

Pre-Condition: **ISPC** application must be installed and the authentication key and password must be set.

Main Scenario*:*

1. User messages the authentication key and password to the smart phone.
2. Smart phone sends back the successful message to the remote mobile.

Alternate Scenario:

1. User messages the wrong authentication key and password to the smart phone.
2. Smart phone does not send back any message to the remote mobile.

**Use case 6:** Disconnection

Primary Actor: User

Pre-Condition: **ISPC** application must be installed and the unauthentication key and password must be set.

Main Scenario:

1. User messages the unauthentication key and password to the smart phone.
2. Smart phone sends back the successful message to the remote mobile.

AlternateScenario:

1. User messages the wrong unauthentication key and password to the smart phone.
2. Smart phone does not send back any message to the remote mobile.

Use case related to Remote Accessing:

**Use case 7:** Contact retrieval

Primary Actor: User

Pre-Condition: Remote phone must be connected to the smart phone. And Contact service must be ON.

Main Scenario:

1. User messages the contact key and the desired contact name.
2. Application receives the contact key and contact name.
3. Application matches the contact key with stored one for correctness.
4. Application searches the contact name in the phone book and sends back the contact detail.

Alternate Scenario:

1. User messages the contact key and the desired contact name.
2. Contact key is wrong then the application will not send any message to the remote phone.

**Use case 8:** Missed call retrieval

Primary Actor: User

Pre-Condition: Remote phone must be connected to the smart phone. And Missed call service must be ON.

Main Scenario:

1. User messages the missed call key.
2. Application receives the missed call key.
3. Application matches the missed call key with stored one for correctness.
4. Application searches the call log for new call and sends back the missed call detail.

Alternate Scenario:

1. User messages the missed call key.
2. Missed call key is wrong then the application will not send any message to the remote phone.

**Use case 9:** Message retrieval

Primary Actor: User

Pre-Condition:Remote phone must be connected to the smart phone. And Message service must be ON.

Main Scenario:

1. User messages the messagekey and the desired name or all.
2. Application receives the message key and name or all.
3. Application matches the message key with stored one for correctness.
4. Application searches the inbox and sends back the message detail back to the remote phone.

Alternate Scenario:

1. User messages the message key and the desired name.
2. Message key is wrong then the application will not send any message to the remote phone.

**Use case 10:** Ringer

Primary Actors:User

Pre Condition: Remote phone must be connected to the smart phone. And Ringer service must be ON.

Main Scenario:

1. User messages the ringer key.
2. Application receives the ringer key.
3. Application matches the ringer key with stored one for correctness.
4. Application makes the smart phone ring even when it is in the silent mode.

Alternate Scenario:

1. User messages the ringer key.
2. Ringer key is wrong, so the application will not ring the smart phone.

**Use case 11:**Profile Changer

Primary Actors: User

Pre Condition: Remote phone must be connected to the smart phone. And Profile changer service must be ON.

Main Scenario:

1. User messages the profile changer key.
2. Application receives the profile changer key.
3. Application matches the profile changer key with stored one for correctness.
4. Application changes the profile of the smart phone from silent to general and vice versa.

Alternate Scenario:

1. User messages the profile changer key.
2. Profile changer key is wrong, so the application will not change the profile of the smart phone.

**Use case 12:**Location Tracker

Primary Actors:User

Pre Condition: Remote phone must be connected to the smart phone. And location tracker service must be ON. Internet facility must be ON.

Main Scenario:

1. User messages the location tracker key.
2. Application receives the location tracker key.
3. Application matches the location tracker key with stored one for correctness.
4. Application sends back the location of the smart phone to the remote phone.

Alternate Scenario:

1. User messages the location tracker key.
2. Location tracker key is wrong, so the application will not send the location of the smart phone.

Use case related to Bluetooth:

**Use case 13:**Bluetooth Remainder

Primary Actors:User

Pre Condition: Bluetooth service must be ON.

Main Scenario:

1. User enters the MAC address of the Bluetooth external device.
2. Application pairs the external device and the smart phone.
3. External Bluetooth device and the smart phone are connected.

Alternate Scenario:

1. User disconnects with the external device.
2. External Bluetooth device and the smart phone are disconnected.

**2.3.2 Non Functional Requirements:**

* Hardware Requirements:
* Smart Phone
* Bluetooth device
* Microcontroller
* Vibrator
* Software Requirements:
* Android tool kit
* MikroC

**Chapter 3**

**SYSTEM DESIGN**

**3.1 Introduction**

This document contains the full description of Design of the “**Intelligent Smart Phone Controller**” application. I have used “Iterative Water Fall model” for the software development. Iterative Water Fall Model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards.

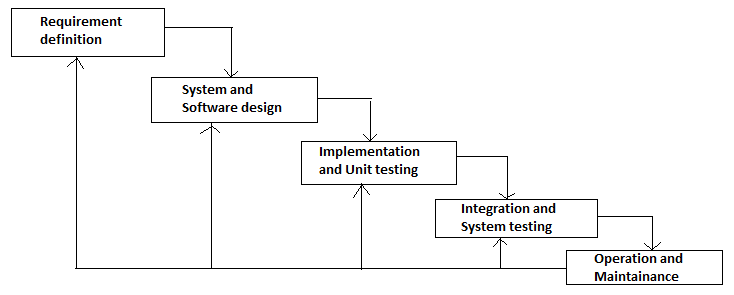


Fig 3.1 Iterative Waterfall model

This document contains the Design of the application according to the modules present in the application. I am using the Object Oriented Modeling and Design methodology which contains the following models,

1. Class diagram
2. Use Case diagram
3. Sequence diagram
4. Activity diagram
5. State Transition diagram

**3.2 Description of UML Diagrams**

**Class diagram:** This provides a graphical notation for modeling class and their relationships, thereby describing possible objects. Class diagrams are useful both for abstract modeling and for designing actual programs. The UML symbol for class is a box.

**Use Case diagram:** This diagram involves a set of use cases and a set of actors. Each use case represents a slice of the functionality the system provides. Each actor represents one kind of object for which the system can perform behavior. The UML notation for use case is an oval and a stick man denotes an actor.

**Sequence diagram:** This diagram shows the participants in an interaction and the sequence of messages among them. A sequence diagram shows the interaction of a system with its actors to perform all or part of a use case.

**Activity diagram:** This diagram shows the sequence of steps that make up a complex process, such as an algorithm or workflow. An activity diagram shows flow of control, similar to a sequence diagram, but focuses on operations rather than on objects.

**State transition diagram:** This diagram shows the flow from one state to another state within the application. A state transition diagram is useful in understanding the transition from one state to another.

**3.3 System Architecture**

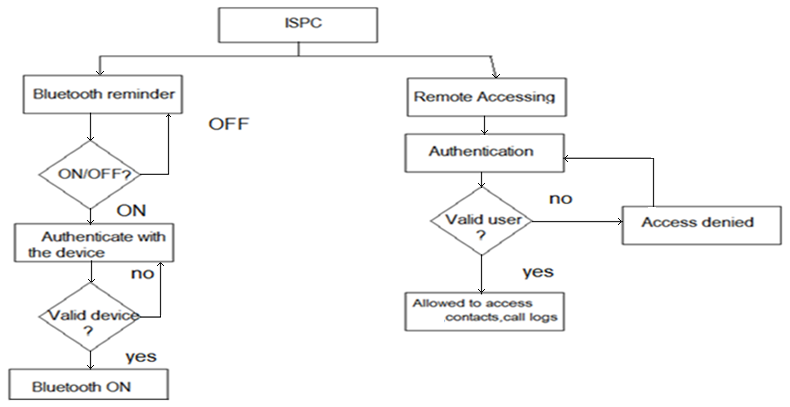


Fig 3.2 System architecture

**3.3.1 Description of the System Architecture:**

After starting the application, two utilities are provided to the user. Oneutility is to turn on or off the Bluetooth remainder and other is to remotely access the smart phone.

In the first section the Bluetooth function of the smart phone is turned ON, then the Bluetooth device and the smart phone are paired (authenticated) and after authenticating the utility is turned ON.

In the second utility which is meant for remote accessing of the smart phone, first the authentication is checked and after correct authentication the user is allowed to access the smart phone remotely from the authenticated phone. The users can access the contacts, messages, call logs, etc. Here the authentication and the accessing are done through the messages.

**3.4 Module Description**

In Intelligent Smart Phone Controller application we the following modules,

* Authentication
* Contact Retrieval
* Missed Call Retrieval
* Message Retrieval
* Ringer
* Profile Changer
* Location Tracker
* Bluetooth Remainder

**3.4.1 Authentication**

This is the first module of the application. In this module the remote user is authenticated. Here authentication is done through a password or secret key. The user sends the password to the smart phone through the remote mobile as a simple SMS. The application in the smart phone checks the password sent by the remote mobile and authenticates it for remote access.

**Class diagram**

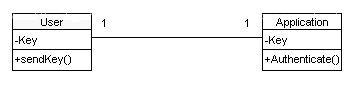


Fig 3.3 Class diagram for Authentication

The class diagram of the authentication module contains two classes, *User* and the *Application.* The relationship between User and Application is *one to one relation.* The Application class has a function to check the key or password.

**Use Case diagram**

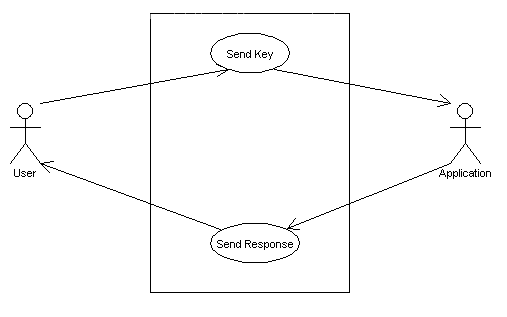
****

Fig 3.4 Use Case diagram for Authentication

The use case diagram of the authentication module consists of two actors *User* and *Application.* User sends the key to the application and the Application receives it and sends the response back to the User.

**Sequence diagram**

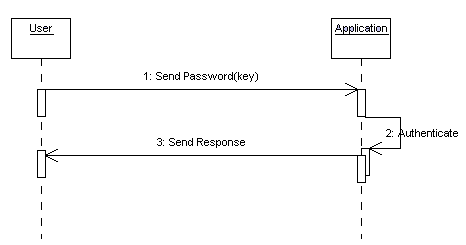
******

Fig 3.5 Sequence diagram for Authentication

Here in the sequence diagram the User sends the password or key to the Application, the Application receives the key and checks for the correctness and sends back the response to the User.

**Activity diagram**

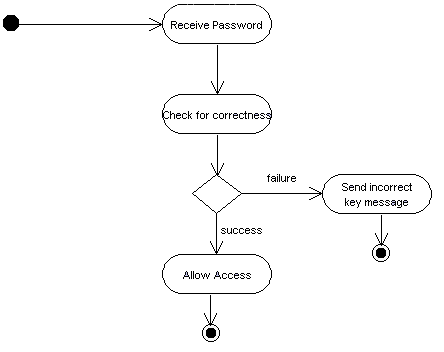
****

Fig 3.6 Activity diagram for Authentication

Here in the activity diagram we see the activities that are carried out in process of authentication. First receive the password from the remote user then check the received password for correctness, if the password is correct then allow the remote user to access the phone and if the password is incorrect then intimate the user that the sent password is wrong.

**State Transition diagram**

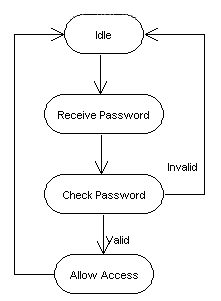


Fig 3.7 State Transition diagram for Authentication

Here we can see different states that application goes while the authentication process is being done. First the application will be in the Idle state, when the password is received it changes from idle to next state and then goes to the password checking state and then if it is valid goes to the remote accessing state and if invalid then again goes back to idle state.

**3.4.2 Contact Retrieval**

After authenticating the remote user, the remote user has an option to retrieve any contacts from his smart phone. In this module all the process related to the contact retrieval is done. Here the remote user has to send a unique key which tell the application that he needs to access the contacts from his smart phone.

**Class diagram**

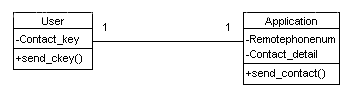


Fig 3.8 Class diagram for Contact Retrieval

In this module we have two class User and the Application. Here the relationship between User and the Application is one to one relationship. Here the key is sent from the User class to the Application class; the Application class receives the key and sends back the required contact detail back.

**Use Case diagram**

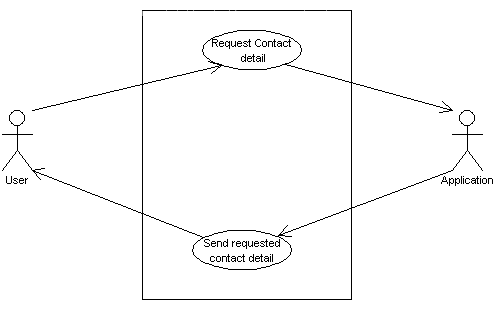


Fig 3.9 Use Case diagram for Contact Retrieval

Here we have two actors User and the Application. The User requests some contact detail and the application searches the requested contact and then sends back the requested contact detail.

**Sequence diagram**

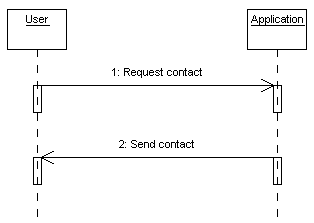


Fig 3.10 Sequence diagram for Contact Retrieval

Here the remote user sends a request for the contact detail and the application checks the contact list in the smart phone and then sends the requested contact back to remote user phone.

**Activity diagram**

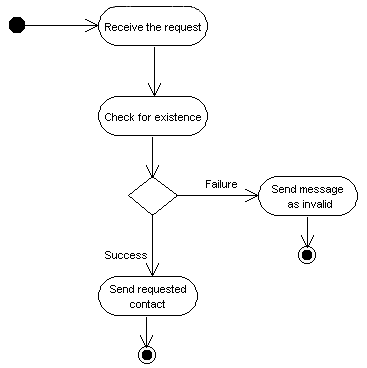


Fig 3.11 Activity diagram for Contact Retrieval

Here the contact request is received by the application and then checks for the existence of the requested data in the contact list. If the requested data is present then the data is sent to remote user phone. If the requested data is not present then the relevant message sent back to the remote user phone.

**State Transition diagram**

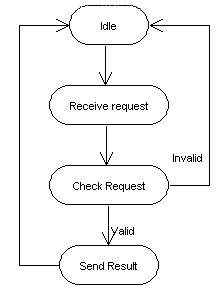


Fig 3.12 State Transition diagram for Contact Retrieval

First the system will in the idle state, then it moves to next state where it will receive the request from the remote user, then after receiving the request it moves to next state where the request is checked for correctness and existence, and then if the requested data is available then it moves to next state where it will send the requested data back to the remote user phone and then to the idle state.

**3.4.3 Missed Call Retrieval**

This module provides the facility to retrieve the missed call log from the smart phone through a remote phone. Here the remote user sends a key to the smart phone through message and the application in the smart phone checks the key and sends the relevant missed call log to the remote user phone.

**Class diagram**

****

Fig 3.13 Class diagram for Missed Call Retrieval

Here we have two classes User and Application. The user sends the key and the application receives the key and checks it and sends the relevant data back to the user.

**Use Case diagram**

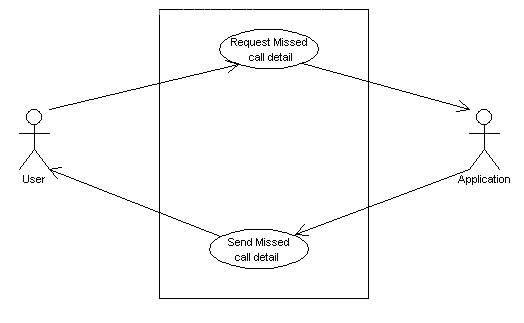


Fig 3.14 Use Case diagram for Missed Call Retrieval

Here we have two actors User and the Application. User requests the missed call log and the Application receives the requests and sends the requested missed call log to the User.

**Sequence diagram**

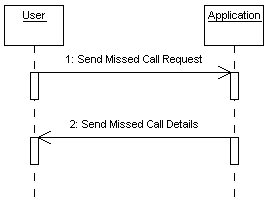


Fig 3.15 Sequence diagram for Missed Call Retrieval

Here the remote user sends a request for the missed call log and the application checks the log list in the smart phone and then sends the requested missed call log back to remote user phone.

**Activity diagram**

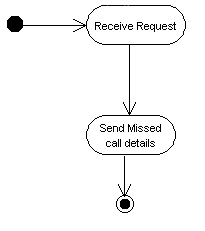


Fig 3.16 Activity diagram for MissedCall Retrieval

Here first the request is received from application and the correctness is checked and then the requested missed call log is sent to the remote user phone.

**State Transition diagram**

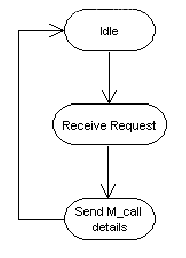


Fig 3.17 State Transition diagram for MissedCall Retrieval

First the application will be in the idle state, when the request is received it goes to receive request state and then it goes to the send missed call log state and sends the requested missed call log to the remote user and then back to the idle state.

**3.4.4 Message Retrieval**

After authenticating the remote user, the remote user has an option to retrieve any contacts from his smart phone. In this module all the process related to the message retrieval is done. Here the remote user has to send a unique key which tell the application that he needs to access the message from his smart phone.

**Class diagram**

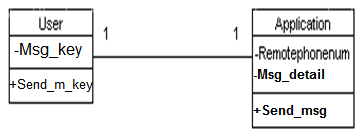


Fig 3.18 Class diagram for Message Retrieval

In this module we have two class User and the Application. Here the relationship between User and the Application is one to one relationship. Here the key is sent from the User class to the Application class; the Application class receives the key and sends back the required unread message back.

**Use Case diagram**

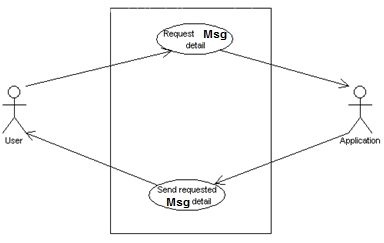


Fig 3.19 Use Case diagram for Message Retrieval

Here we have two actors User and the Application. The User requests unread message and the application checks whether there are any new message and then sends back the unread message.

**Sequence diagram**

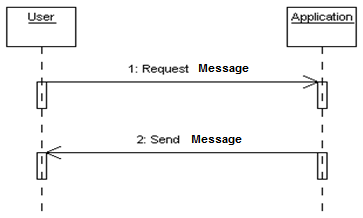


Fig 3.20 Sequence diagram for Message Retrieval

Here the remote user sends a request for unread messages and the application checks the inbox in the smart phone and then sends back the unread message to remote user phone.

**Activity diagram**

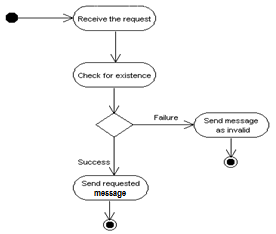


Fig 3.21 Activity diagram for Message Retrieval

Here the unread message request is received by the application and then checks for the existence of the requested data in the message inbox. If the requested data is present then the data is sent to remote user phone. If the requested data is not present then the relevant message sent back to the remote user phone.

**State Transition diagram**

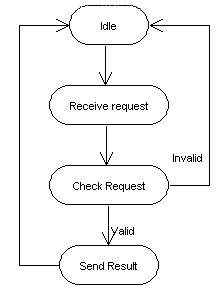


Fig 3.22 State Transition diagram for Message Retrieval

First the system will in the idle state, then it moves to next state where it will receive the request from the remote user, then after receiving the request it moves to next state where the request is checked for correctness and existence, and then if the requested data is available then it moves to next state where it will send the requested data back to the remote user phone and then to the idle state.

**3.4.5 Ringer**

This module provides the user to make the smart phone ring when it is in the silent mode. Here the remote user sends a key through message to the smart phone and the application receives it and makes the smart phone ring when it is in the silent mode.

**Class diagram**

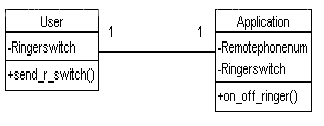


Fig 3.23 Class diagram for Ringer

Here this module has two classes User and Application. A key is sent through the user and the application receives it. In Application class the correctness is of key is checked and the ringer is set ON/OFF.

**Use Case diagram**

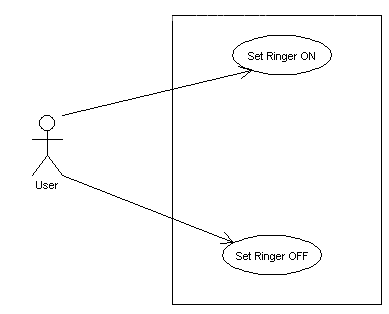


Fig 3.24 Use Case diagram for Ringer

In this module we have only one actor User. He sends the key to set the ringer ON/OFF. The key is predefined and only the correct key will be answered by the application.

**Sequence diagram**

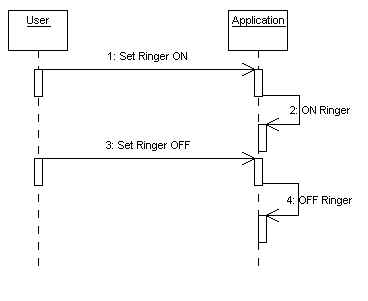


Fig 3.25 Sequence diagram for Ringer

Here the remote user sends the unique key to the smart phone and the application receives it and sets the requested option, i.e. ringer ON or ringer OFF.

**Activity diagram**

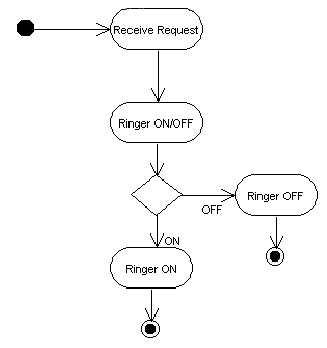


Fig 3.26 Activity diagram for Ringer

Here the first activity is initiated when the request is received by the application. In the next activity it checks whether the request is to turn ON the ringer or turn OFF the ringer and accordingly it moves to the further activity.

**State Transition diagram**

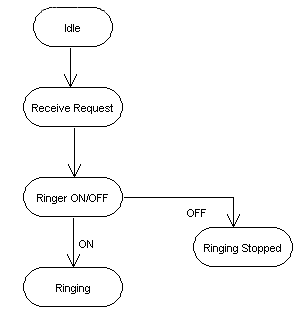


Fig 3.27 State Transition diagram for Ringer

First the application will be in the idle state, when the request is received it moves to receive request state and then according to the request it moves to ringing state or ringer off state.

**3.4.6 Profile Changer**

This module provides the remote user to change the profile setting of the smart phone. The application in the smart phone takes the key from the remote user and changes the profile as needed. The key from the remote user is sent as a simple message.

**Class diagram**

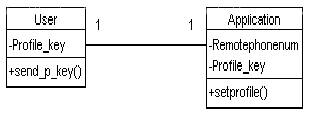


Fig 3.28 Class diagram for Profile Changer

Here we have two classes User and Application. The remote user sends the key to the application as a simple message and the application receives it and changes profile as needed through set profile function.

**Use Case diagram**

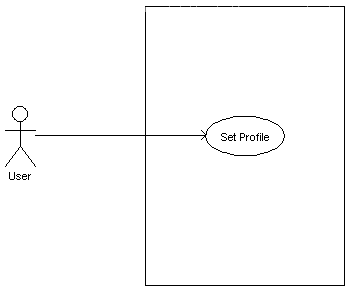


Fig 3.29 Use Case diagram for Profile Changer

Here we have only one actor he is the User or the remote user. He sends the request to set the profile to particular setting; the application receives it and sets the profile.

**Sequence diagram**

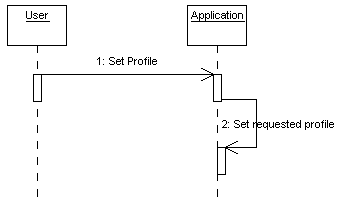


Fig 3.30 Sequence diagram for Profile Changer

Here the action starts when the remote user sends request for profile change, the application validates the request and sets the desired profile in the smart phone.

**Activity diagram**

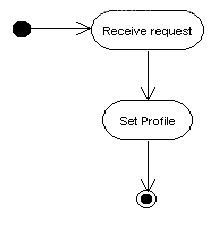


Fig 3.31 Activity diagram for Profile Changer

Here when the request is received then the activity starts, first it will be receive request activity and then it moves to set profile activity. In the second activity the requested profile is set.

**State Transition diagram**

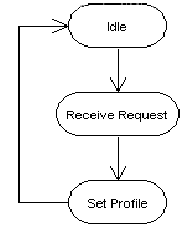


Fig 3.32 State Transition diagram for Profile Changer

First the application will be in the idle state, when the request is received it moves to receive request state. After receiving the request it moves to set profile state and the desired profile is set and then it moves back to the idle state.

**3.4.7 Location Tracker**

This module provides the remote user an option to find the location of the smart phone. Here the location is sent as the longitude and latitude. The information is sent to the remote user phone as a simple message. By using this longitude and latitude in Google maps we can identify the location of the smart phone.

**Class diagram**

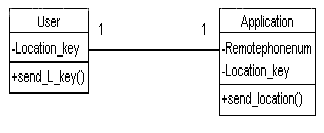


Fig 3.33 Class diagram for Location Tracker

Here we have two classes User and Application. The relation between the two classes is one to one relationship. A key is sent to the application and the application in return sends the location of the smart phone to the remote user.

**Use Case diagram**

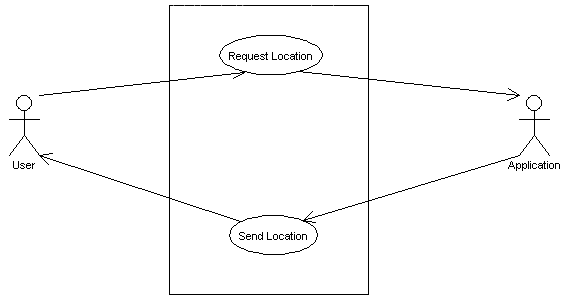


Fig 3.34 Use Case diagram for Location Tracker

Here we have two actors User and Application. User requests for location of the smart phone and the Application sends the location to the User.

**Sequence diagram**

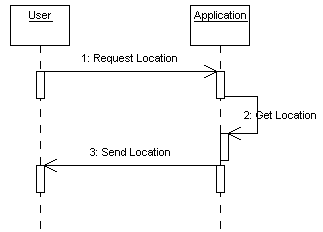


Fig 3.35 Sequence diagram for Location Tracker

Here the User sends the request to get the location of the smart phone to the application. Application gets the location from the system and then sends back the location to the remote user.

**Activity diagram**

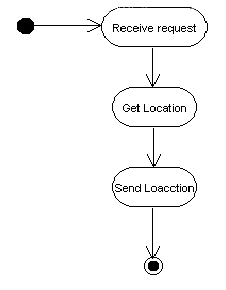


Fig 3.36 Activity diagram for Location Tracker

Here the activity starts when the request from the remote user is received by the application. After receiving it moves to the get location activity and then it moves to the send location activity where it will send the retrieved location to the remote user.

**State Transition diagram**

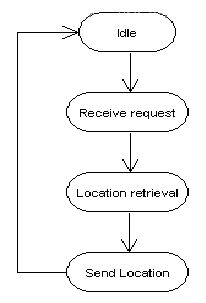


Fig 3.37 State Transition diagram for Location Tracker

First the application will be in the idle state, when the request is received by the application it moves to the receive request state. After receiving the request it moves to the location retrieval state and then to the send location state, and at last it again moves back to idle state.

**3.4.8 Bluetooth Remainder**

This module provides the user to ON/OFF the Bluetooth remainder option. Here to turn ON the functionality first the smart phone will pair with the Bluetooth device and then it turns ON the function. When the user selects to turn OFF the function then the connectivity between the Bluetooth device and the smart phone is disconnected.

**Class diagram**

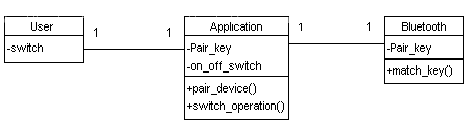


Fig 3.38 Class diagram for Bluetooth Remainder

Here we have three classes User, Application and Bluetooth. The relation between user and application is one to one relationship. The relation between application and Bluetooth is also one to one relationship.

**Use Case diagram**

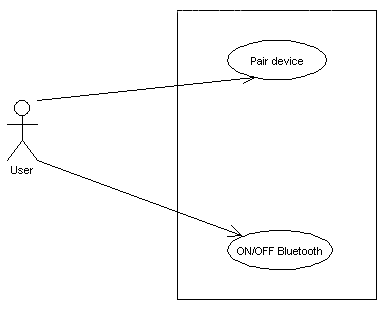


Fig 3.39 Use Case diagram for Bluetooth Remainder

Here we have only one actor User. User turns ON/OFF the Bluetooth function and after turning ON the smart phone is paired with the external Bluetooth device.

**Sequence diagram**

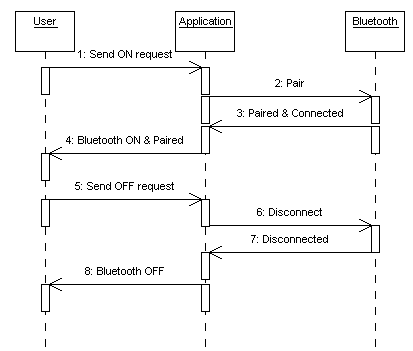


Fig 3.40 Sequence diagram for Bluetooth Remainder

Here when the user turns ON the function the application will pair with the Bluetooth device and then sends the confirmation to the application that it is connected. When the user turns OFF the function the application disconnects with the device and returns the result to application.

**Activity diagram**

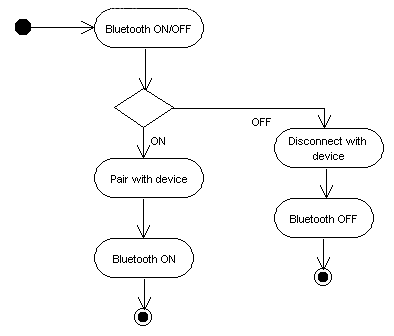
****

Fig 3.41 Activity diagram for Bluetooth Remainder

Here the first activity is to turn ON/OFF the Bluetooth function. If the user selects to turn ON the function then it moves to next activity where the device is paired and then the function is set ON. If the user selects to turn OFF the function then it moves to disconnecting device activity and then the function is turned OFF.

**State Transition diagram**

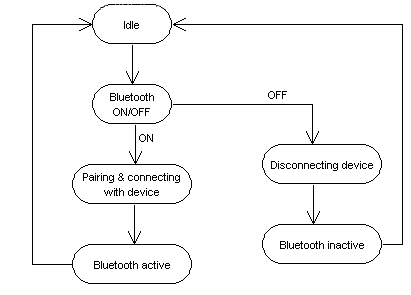
****

Fig 3.42 State Transition diagram for Bluetooth Remainder

First the application will be in the idle state. When the user selects to turn ON/OFF the Bluetooth functionality it moves to next state. If the user wants to turn ON then it moves to the pairing and connecting state and then back to idle state. If the user wants to OFF the functionality then it moves to disconnecting state and then back to idle state.

**Conclusion**

It can be very useful for the users to be in touch with their mobile. It can reduce the chances to lose the mobile and can also be very useful to get certain information when the mobile is not with the user.

FUTURE ENHANCEMENT

* Access all messages received from a specific user both read and unread messages
* Application should send error messages if wrong key is given
* Using IME number to locate the device

**BIBILIOGRAPHY**

* **Books Referred**
* Head First Android Development by Jonathan Simon
* Android Application Development By O'reily
* Android Programmer's Guide By Jerome Dimarzio.
* Android Application Development ByReto.
* **Web site Referred**
* <http://market.android.com>
* <http://developer.android.com/guide/basics/what-is-android.html>
* [http://developer.android.com/reference/packages html](http://developer.android.com/reference/packages%20html)
* books.google.com/help/eBooks/android.html
* <http://developer.android.com/reference/android/telephony/PhoneStateListener.html>
* <http://developer.android.com/reference/android/widget/package-summary.html>