Chapter1

**INTRODUCTION \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* 1. **INTRODUCTION ABOUT THE VOICE SMS**

Voice SMS is an application developed in this work that allows a user to send spoken messages into SMS text message and vice versa. User can send messages to the entered phone number or the number of contact from the phonebook. Speech recognition is done via the Internet, connecting to Google's server. The application is adapted to input messages in English. Used tools are Android SDK and the installation is done on mobile phone with Android operating system.

The application "**voice sms**" is an application developed for android based Operating System employed in mobile phones. This application is developed to read and send text messages and spell out through voice.

Mobile marketing gains more popularity as a number of mobile device users are on the increase the ways of doing businesses and marketing have tremendously changed from the past. Forms of businesses using mobile marketing include mobile advertising, games, online stores, and location-based services, order tracking, ticket purchases. Medical field is no exception. The prime objective of “ **VOICE SMS**” is to help disabled people who are unable to type text messages like visually impaired people.

Benefits of Voice SMS Service:

• Personal Touch: Voice message has greater emotions compared to text; hence a more personal way of communicating

• Reach out to large Indian target audience which is not well versed in English language

• VOICE SMS is handset or SIM independent, hence can be used universally for all

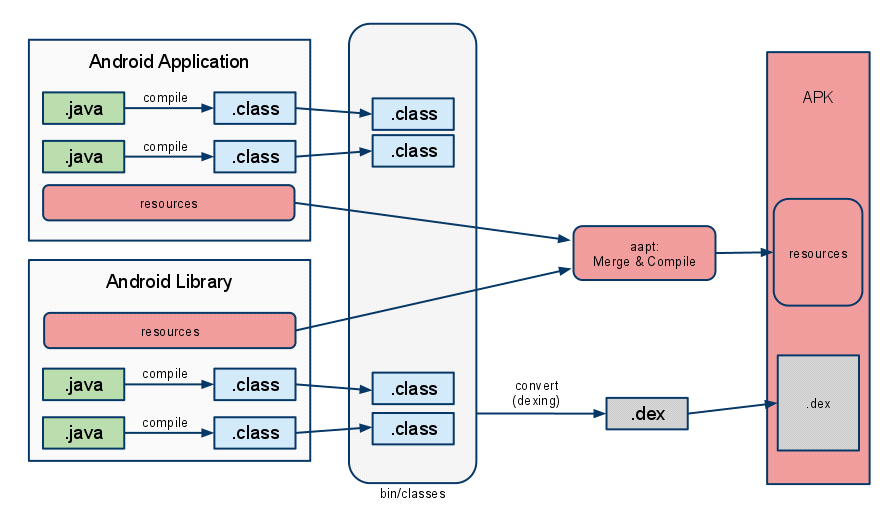
Mobile users.

.

**1.2 SUPPORTING SYSTEM**

**Overview of Android**

The Android operating system (OS) has come a long way since the announcement of the Open Handset Alliance in late 2007.The idea of an open source OS for embedded systems was not new, but Google aggressively backing it definitely has helped push Android to the forefront in just a few years.



**Fig 1.1: Java Android Flow execution**

Many wireless carriers in multiple countries across various communication protocols have one or more Android phones available. Other embedded devices, such as tablets, net-books, televisions, set-top boxes, and even automobiles, have also adopted the Android OS.

**The Evolution of Android**

Google, seeing a large growth of Internet use and search in mobile devices, acquired Android, Inc., in 2005 to focus its development on a mobile device platform. Apple introduced the iPhone in 2007 with some ground-breaking ideas including multi touch and an open market for applications. Android was quickly adapted to include these features and to offer definite distinctions, such as more control for developers and multitasking. In addition, Android incorporates enterprise requirements, such as exchange support, remote wipe, and Virtual Private Network (VPN) support, to go after the enterprise market that Research In Motion has developed and held so well with its Blackberry models.

Device diversity and quick adaptation have helped Android grow its user base, but it comes with potential challenges for developers. Applications need to support multiple screen sizes, resolution ratios, keyboards, hardware sensors, OS versions, wireless data rates, and system configurations. Each can lead to different and unpredictable behaviour, but testing applications across all environments is an impossible task.

Android has therefore been constructed to ensure as uniform an experience across platforms as possible. By abstracting the hardware differences, Android OS tries to insulate applications from device-specific modifications while providing the flexibility to tune aspects as needed. Future-proofing of applications to the introduction of new hardware platforms and OS updates is also a consideration. This mostly works as long as the developer is well aware of this systematic approach. The generic Application Programming Interfaces (API) that Android offers and how to ensure device and OS compatibility a remain threads discussed throughout. Still, as with any embedded platform, extensive testing of applications is required.

Google provides assistance to third-party developers in many forms as Android Development Tool (ADT) plugins for Eclipse (also as standalone tools) including real-time logging capabilities, a realistic emulator that runs native ARM code, and in-field error reports from users to developers of Android Market applications.

**Google and the Open Handset Alliance**

In 2007, a group of handset manufacturers, wireless carriers, and software developers (notably, Google) formed the Open Handset Alliance, with the goal of developing the next generation of wireless platform. Unlike existing platforms, this new platform would be non-proprietary and based on open standards, which would lead to lower development costs and increased profits. Mobile software developers would also have unprecedented access to the handset features, allowing for greater innovation.

As proprietary platforms such as RIM BlackBerry and Apple iPhone gained traction, the mobile development community eagerly listened for news of this potential game-changing platform.

**Cheap and Easy Development**

If there’s one time when “cheap and easy” is a benefit, it’s with mobile development. Wireless application development, with its ridiculously expensive compilers and preferential developer programs, has been notoriously expensive to break into compared to desktop development. Here, Android breaks the proprietary mold. Unlike with other mobile platforms, there are virtually no costs to developing Android applications. The Android SDK and tools are freely available on the Android developer website, http://developer.android.com. The freely available Eclipse program has become the most popular integrated development environment (IDE) for Android application development; there is also a powerful plug-in available on the Android developer site for facilitating Android development.

So we’ve covered cheap; now let’s talk about why Android development is easy. Because Android applications are written in Java, developers will be familiar with many of the packages provided as part of the Android SDK, such as java, .net . Developers will be pleased .

**1.3 SCOPE**

Scope of the project is to create a mobile application is now days many smart messengers are available in Android market which are based on voice to text transmission. Sometimes we call it as instance messenger in that ,if you want to send message, speak that message, message will be converted into text and sent as SMS. In many cases these messengers provide option for sending message, not for received messages. These applications can be used by visually impaired people.

* 1. **OBJECTIVE**

The main objective of the project is to create android application Messaging System, which is Voice enabled. The application listens to your messages and thenresponds with voice commands by talking. The application converts your text into voice and voice into text. For Android it is Voice- to- Text technology to listen to what you send and gets youconnected with people.

**1.5 PURPOSE**

Our project’s aim is to help the differently abledpeople to interact with other’s through our application. The visually impaired people, many times find it difficult tointeract with other people through current messaging system.The application provides with better user interface and interaction is completely through voice, where the user does not need to use the physical touch or press any key for interaction. As the application is built on top of the SMS layer,

so there is no need of installing application at both the ends. Also, the application provides facility to read message in multiple languages.

Chapter 2

**REVIEW OF STATE OF ART \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The user speaks message and contact number as stated using voice API we convert the voice into text an vice versa to voice Proposed system reduces these unnecessary efforts of adapting to hand sets and to help .The application listens to your messages and then responds with voice commands by talking. The application converts your text into voice and voice into text. For Android it is Voice- to- Text technology to listen to what you send and gets you connected with people.

User can send messages to the entered phone number or the number of contact from the phonebook. Speech recognition is done via the Internet, connecting to Google's server. The application is adapted to input messages in English. Used tools are Android SDK

Android has therefore been constructed to ensure as uniform an experience across platforms as possible. By abstracting the hardware differences, Android OS tries to insulate applications from device-specific modifications while providing the flexibility to tune aspects as needed. Future-proofing of applications to the introduction of new hardware platforms and OS updates is also a consideration

This application is developed to read and send text messages and spell out through voice.

Chapter 3

**PROBLEM DEFINITION**

Object–oriented analysis is concerned with developing software engineering requirements and specifications that expressed as a system’s abject, as opposed to the traditional data or functional views of systems. It is the process of defining the problem in terms of objects: real-world objects with which the system must interact and candidate software objects used to explore various solution alternatives. The natural fit of programming objects to real-world objects has a big impact here. Maintainability, reusability, productivity are the benefits yielded from the Object –oriented Analysis

**3.1 EXISTING SYSTEM**

Incorporating speech recognition and text-to-speech ([TTS](http://www.webopedia.com/TERM/T/TTS.html)) services may prove very beneficial to certain types of applications and certain groups of users. Speech recognition involves listening for user's voice input, processing the recorded sound, and interpreting the results. TTS services involve taking text string data and having the device "read" the content aloud using a "voice". Hands-free applications, such as turn-based navigation utilities routinely use both technologies. Users with special needs, such as the visually impaired, also benefit from these features.

Android speech services are available within the SDK in the [*android.speech*](http://developer.android.com/reference/android/speech/package-summary.html) package. The speech recognition classes, such as the definition of the RecognizerIntent, can be found within this package. The TTS features are found in the [*android.speech.tts*](http://developer.android.com/reference/android/speech/tts/package-summary.html) sub-package.

Applications require no special permissions to use Android speech services. Be aware, though, that speech recognition does require a data connection.

**3.2 PROBLEM STATEMENT**

The application provides with better user interface and interaction is completely through voice, where the user does not need to use the physical touch or press any key for interaction. As the application is built on top of the SMS layer and to take notes setting reminderand we are developing further to include in further updates.

**3.3 PROPOSED SYSTEM**

Voice SMS is a mobile based application which is used to interact with users on mobile. using messages. We included Google Voice Recognition Activity instead of traditional way of sending messages by typing on different keyboard layouts .The user speaks message and contact number as stated using voice api we convert the voice into text an vice versa to voice Proposed system reduces these unnecessary efforts of adapting to hand sets and to help

Chapter 4

**SYSTEM REQUIREMENTS AND ANALYSIS**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Software Requirement specification is the starting point of the software developing activity. As system grew complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the needs for the requirement phase arise. The software project is initiated by the client needs. The software requirement specification is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase).

**4.1 FUNCTIONAL REQUIREMENTS**

* Creating a User Interface using Android for demonstrating E-Prescription Application
* It consists of patient registration, login and doctors login , homepage interfaces.
* The process involves creating Writing Module, viewing the details of selected patient, viewing previous request status, MakeRequset.

**4.2 NON FUNCTIONAL REQUIREMENTS**

* ***Maintainability****:*All the modules must be clearly separate to allow different user interfaces to be developed in future. Through thoughtful and effective software engineering, all steps of the software development process will be well documented to ensure maintainability of the product throughout its life time. All development will be provided with good documentation.
* ***Performance****:*The response time, utilization and throughput behaviour of the system. Care is taken so as to ensure a system with comparatively high performance.
* ***Usability****:*The ease of use and training the end users of the system is usability. System should have qualities like- learning ability, efficiency, affect, control. The main aim of the project is to increase the scope of page designer to design a page and to reduce the rework of the programmer.
* ***Modifiability****:*The ease with which a software system can accommodate changes to its software is modifiability. Our project is easily adaptable for changes that is useful for the application to withstand the needs of the users.
* ***Portability****:*The ability of the system to run under different computing environments. The environment types can be either hardware or software, but is usually a combination of two.
* ***Reusability****:* The extent to which existing application can be reused in new application. Our application can be reused a number of times without any technical difficulties.
* ***Security****:*The factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Security can be ensured as the project involves authenticating the users.

**4.3 SOFTWARE REQUIREMENTS**

* *Software Requirements for system*
* Language:J2ME 1.1, XML
* Operating System: Windows 7.
* SDK: Android SDK.
* Design: Rational Rose Enterprise edition.

*ii. Software Requirements for Mobile*

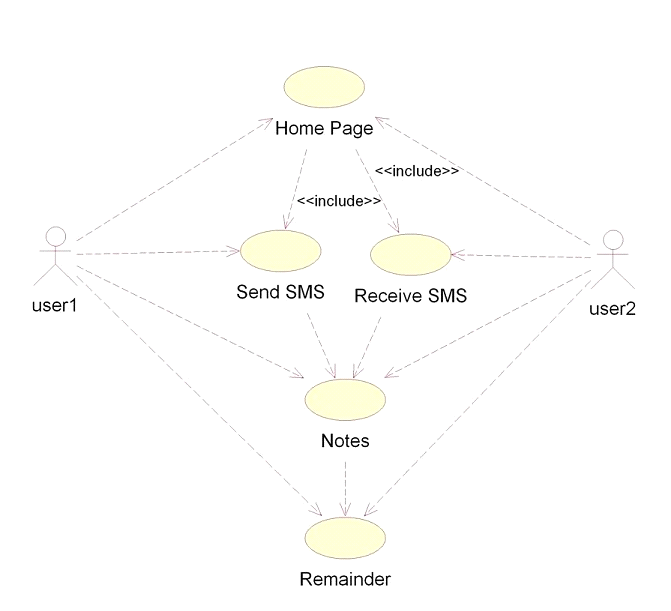
* Android version : minimum of 2.2

**4.4 HARDWARE REQUIREMENTS**

* *Hardware Requirements for system*
* System Processor :Intel Core 2 Duo
* System speed : 1.2 GHz
* System hard disk: 80 GB
* System RAM: 2GB
* *Hardware Requirements for Mobile*
* Mobile processor: Cortex A5 Single core
* Mobile speed: 1 GHz
* Mobile RAM : 512 MB
* Mobile internal storage:1 GB

**4.5 USE CASE ANALYSIS**

Usecase diagram is one among the 9 UML diagrams.The usecase diagrams provides us the pictorial representation of each and every individual involved in the application.It involves actors and their actions.



**Fig 4.1: Use case Diagram for Voice SMS**

**Use Case Descriptions:**

The brief descriptions of the actions performed by an actor are as follow

**Table 4.1: Use Case Description for Voice SMS app**

|  |  |
| --- | --- |
| Description | This page is used to display app home page |
| Actor | User1,user2 |

**Table 4.2: Use Case Description for Send SMS**

|  |  |
| --- | --- |
| Description | In this user sends the sms |
| Actor | User1,Voice SMS app |
| Excepted input | Contact number,body of the message |
| Excepted output | If the enter details are valid then notify user and send sms |

**Table 4.3: Use case Description for Receive SMS**

|  |  |
| --- | --- |
| Description | In this the app speaks out the received message |
| Actor | User2 ,Voice SMS app |
| Excepted output | The received message is converted into speech form using TTS and then with the help of mobile audio hardware the message is spoken out. |

**Table 4.4: Use case Description for Make Note**

|  |  |
| --- | --- |
| Description | The notes from user are saved. |
| Actor | Users , Voice SMS app |
| Excepted input | Note title , content of note |
| Excepted output | If the input is not null display the saved note along with previous notes. |

**Table 4.5: Use case Description for Remainder**

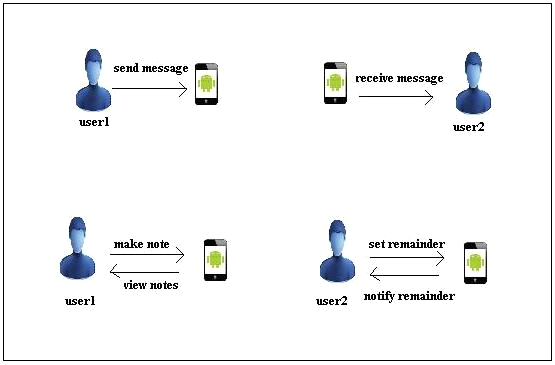
|  |  |
| --- | --- |
| Description | This is used to display the remainders if any. |
| Actor | Users, Voice SMS app |
| Excepted input | Users text content |
| Excepted output | Display remainders which are set to a particular time. |

Chapter 5

**SYSTEM ARCHITECTURE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**5.1 SYSTEM ARCHITECTURE AND ITS DESCRIPTION**



**Fig 5.1 System Architecture for Voice Sms**

First of all we need to install **"Voice Sms APP**" on the android device.Now the user opens the app, there will be different fields for sending and receiving SMS, note making and setting remainder fields.Then user deligates and choose any of the available options .if the user wants to send message then he clicks on speak icon and input their voice and sends it. Now, the sent SMS is received by google API and it decodes the received voice SMS and converts into appropriate text message and sends it to the desired user. Now, the desired user receives the text message. The app will recognize the text and converts into appropriate voice message using text to speech mechanisim and speaks out the text message as voice output.

**5.2 MODULES DESCRIPTION**

**5.2.1 SMS sending Module**

**Description and Priority**

The user needs to enter the contact name and message in order to send the message after completion of entering the details. The module-1 has first and high priority.

**MODULE-1:-**

This module deals with message sending where the user has to provide contact number and will be authenticated.

**Stimulus/Response Sequences**

In this stimulus sequences it check whether the user is authentic or not and if the user is authentic it allows the user to send a message , else if the user is unauthorized then displays an alert stating the user are invalid.

**MODULE-2:-**

**SMS receiving Module**

In this stimulus sequence it receives the message along with contact number as input and check whether the user details are correct and finally display the received message.

Chapter 6

**SYSTEM DESIGN**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**6.1 CLASS DIAGRAM:**

Class diagram are used to describe the overall structure of the system. Classes are abstractions that specify the structure and behaviour of set of objects. Object are instances of classes that are created, modified and destroyed during the execution of the system. Class diagrams describe the system in terms of objects, classes, attributes, operations and their associations**.**

**Class Diagram Voice SMS :**

**Fig 6.1: Class Diagram for Voice SMS**

In this system we have four classes called send message activity, reminder activity,

In this class diagram, these classes are represented with boxes which contain three parts.

* The upper part holds the name of the class
* The middle part contains the attribute of the classs
* The bottom part gives the methods or operations the class can take or undertake

**Send Message Class:**

This class contains the attributes as, contact no. and body

Contains the following Methods:

Void sendMessage( )

Void checkTTS( )

initilizesmsReceiver( )

registersmsReceiver( )

**Reminder Class:**

In this class,it contains the attributes as name and time.

Contains the following Methods:

setReminder( )

deleteReminder( )

**Note Class:**

In this class,it contains the attributes as title and body.

Contains the following Methods:

makeNote( )

readNote( )

deleteNote( )

**Receive Message Class:**

In this class it contains the attributes as contact no, title and body, content, audio1 and audio2.Contains the following methods

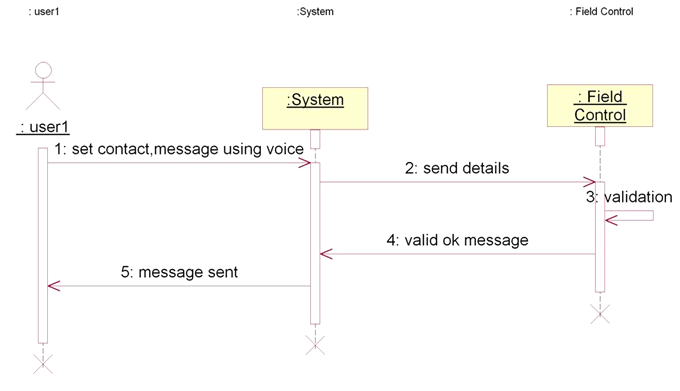
readMessage ( )

speak( )

**6.2 SEQUENCE DIAGRAM**

Sequence diagram is one kind of interaction diagrams, which shows aninteraction among a set of objects and their relationships (another kind of interaction diagram is collaboration diagram). The purpose of the Sequence diagram is to document the sequence of messages among objects in a time based view. The scope of a typical sequence diagram includes all the message interactions for (part-of) a single use case. This diagram addresses a dynamic view objects.

**Sequence diagram for Message sending :**

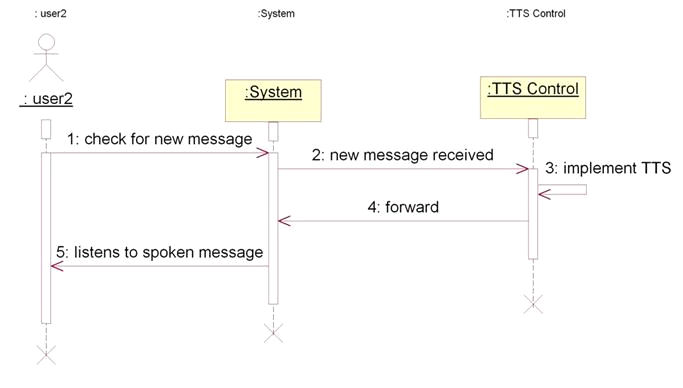


**Fig 6.2 Sequence diagram for message sending**

**Description**:

In this, first the user1 needs to enter the contact and input message through voice and send to system ie, to the application. Once entered data is valid ,then the message is sent sucessfully. If any of the fiels such as contact number fied , message field are empty or both the fields are empty then message sending will be failed..

**Sequence Diagram for message receiving :**

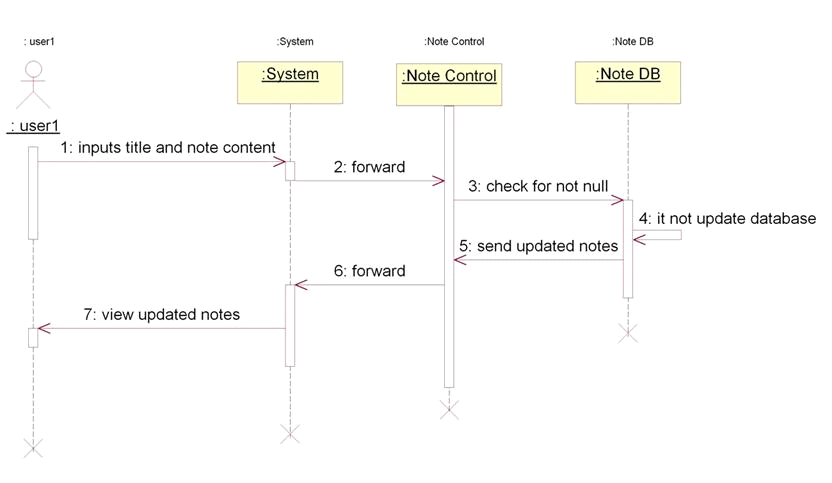


**Fig 6.3 Sequence Diagram for message receiving**

**Description:**

In this sequence diagram the user2 checks the app if any new message is received or not. if any new message is received the the TTS in the app will automatically converts the received text into voice forwards it to the app then the user will listens to the voice message.

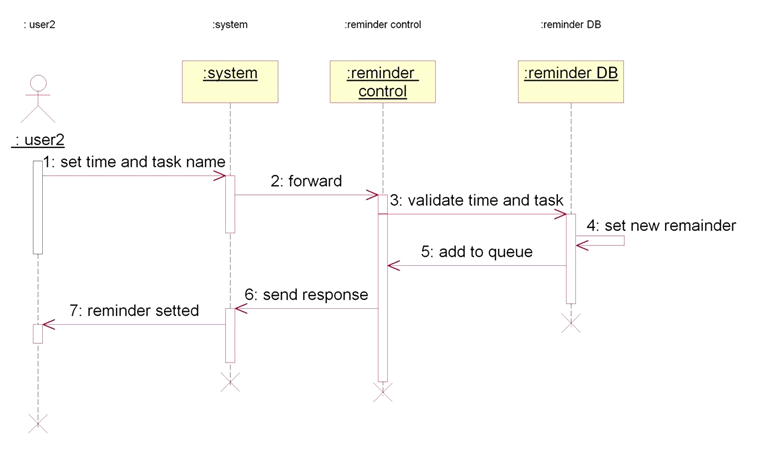
**Sequence Diagram for Make note :**

 **Fig 6.4: Sequence Diagram for Make note**

**Description:**

In this sequence diagram, the user will inputs the note tittle and content and forwards it to the app. The note control field will check the input fields if it is not null it wont take the data if ther is any data it woll be updated and displays the updated note to the user.

**Sequence diagram for creating a reminder :**



**Fig 6.5 Sequence diagram for creating reminder**

**Description**:

In these sequence diagram,the user will set the time and task name in the remainder field and and send it to the system.the system will send the entered data to the remainder control. The remainder control will validate time and task and saves the data in remainder database.now the data will be added to the remainder queue and it as response to the app so that user can view it.

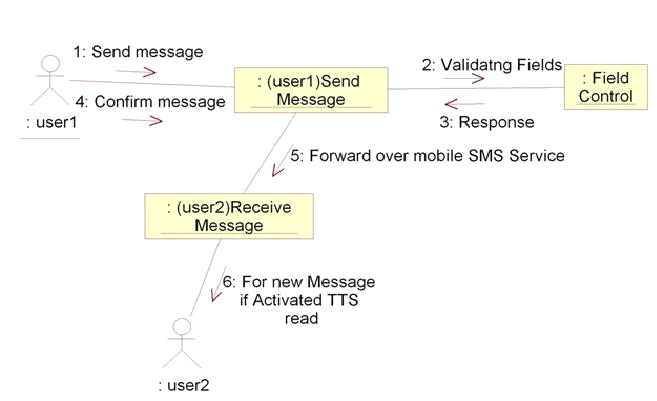
**6.3 COLLABORATION DIAGRAM:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time.

Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message

**Collaboration diagram for Sending message and receiving messsage**

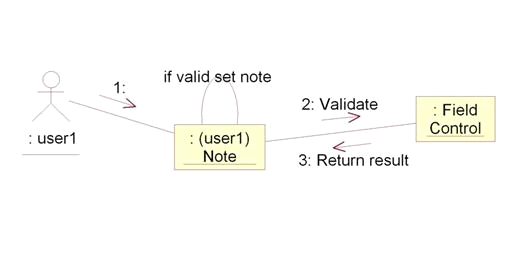


**Fig 6.6: Collaboration diagram for message sending and receiving**

**Description:**

In this diagram, the user speaks the contact anumber and along with followed by their voice message. The message control field will check the fields if they are empty the message sending will be failed.after successful sending of voice message the user2 has to switch on the app. the the app will automatically receives the sent SMS from the sender and now the the TTS mechanisim in the google API will invoke the received message content and converts it into the speech form now the user have to press “activate message speaking” button to listen to message. received message.

**Collaboration diagram for making note:**



**Fig 6.7: Collaboration diagram for message sending and receiving**

**Description:**

In this diagram the user as to first set the note title and then they have to type the note content.if the the fields are valid then the note content will be saved in mobile storage memory. It can be displayed whenever the user wants to view it.

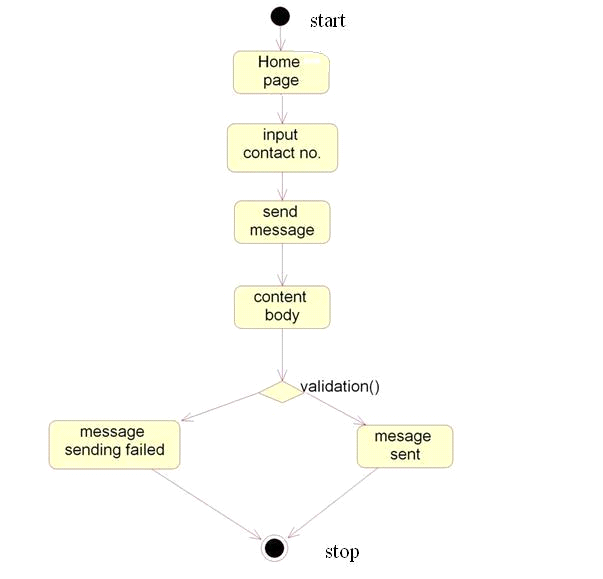
**6.4 STATE-CHAT DIAGRAM**:

State chart diagram is one of the five UML diagrams used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So Statechart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So the most important purpose of Statechart diagram is to model life time of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system

**State chart Diagram for Sending SMS :**

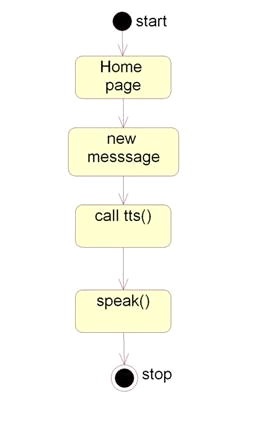


**Fig 6.8 State chart Diagram for Sending SMS**

**Description:**

In this diagram first user opens the app then he will see a home page which contains the fields like input contact no and message body fields.user speaks the contact no. and their voice as message and sends it to the desired user.If the fields are empty then message sending will be failed, else message will be sent successfully.

**State chart Diagram for Received SMS :**

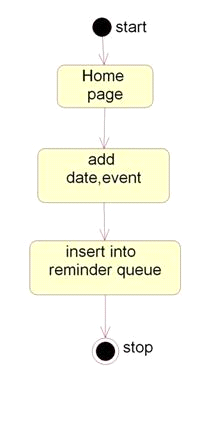


**Fig 6.9 State chart Diagram for Received SMS**

**Description:**

In this diagram, process begins by installing the app and click on that,it diaplays home page.Next pthe user will create a new message and sends it by inputting the contact no. and clicks the speak button , now tts mechanisim in the app will invoke and receives the voice of the user and sends it to the desired user. First of all the user needs to turn on the data connection to enable the app to work for sending the voice to the googe API.

**State Chart Diagram for making Reminder:**



**Fig 6.10: State Chart Diagram making Reminder**

**Description:**

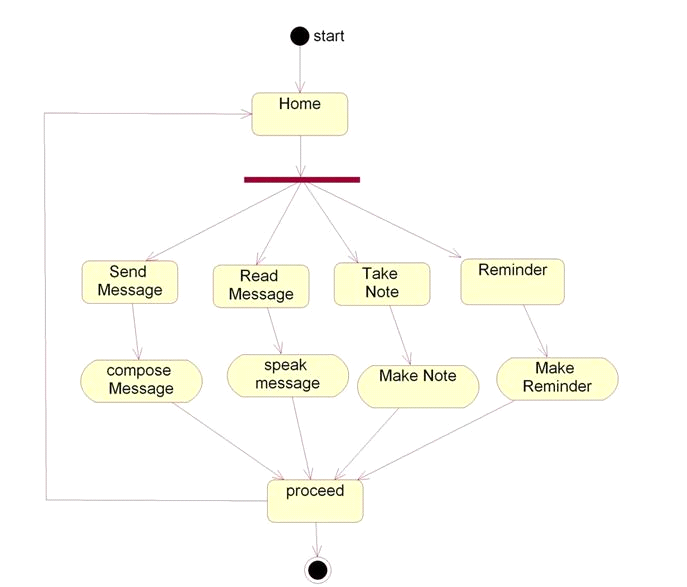
In this diagram the user will first enter into the homepage and check for the reminders and he can also adds the reminders by inputting the time and reminder content through text.now the typed text will be saved and will be displayed at particular time to reminder setted time.

**6.5 ACTIVITY DIAGRAM:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity.

The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork,join etc.

**Activity Diagram for Voice SMS :**



**Fig 6.11: Activity Diagram for Voice SMS**

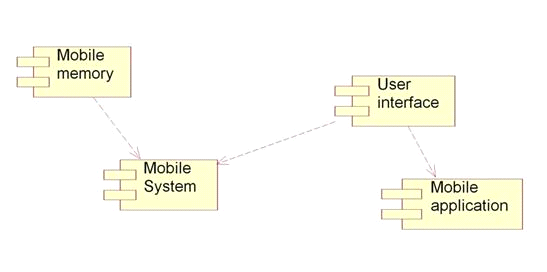
**Description:**

In this activity first we install the app and turn on data connection for sending message,for receiving the message the user no needs to turn on the internet. click on Voice SMS app then it displays the App home page .In that main menu list will appear.The list contains following options as. Contact no.message body, SMS content field, note making and reminder fields.we need to click on contact no. fiels and input the contact no. through voice and speak out the message and press send button.Thevoice message will be read by TTS and it will forward it to the google Api now the Api will convert the voice message into appropriate text and sends it to the desired contact no. user. The app also contains features such asnote making and reminder setting facilities.The user needs to input date for creating any reminder.

**6.6 COMPONENT DIAGRAM:**

Component diagram provide a physical view of the current model. A component diagram shows the oraganization and dependencies among software component,and executable components.

The diagrams also show the externally-visible behavior of the component by displaying the interfaces of the components. Calling dependencies among components are shown as dependency relationships between components and interfaces on other components. Note that the interfaces acutually belong to the logical view, but they can occur both in class diagrams and in component diagrams. Component diagram contain component packages, components, interfaces dependency relationships.



**Fig 6.12: Component Diagram for Voice SMS**

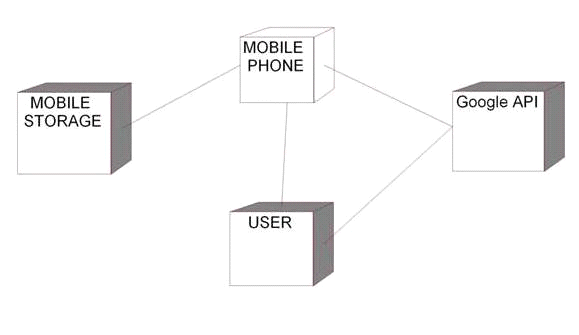
**Description:**

In this component diagram , we have four components they are Mobile memory, user Interface,Mobile System and mobile application.In this the mobile memory has dependency relationship on mobile system component and user interface has dependency relationship in mobile application component.

**6.7 DEPLOYMENT DIAGRAM:**

A deployement diagram shows the configuration of run time processing nodes and the components that live on them. Deployement diagrams address the static deployement view of architecture.

A node typically hosts one or more articrafts. We can import and generalization relationships between packages. Graphically a deployement diagram is a collection of vertices and arcs. A Deployement diagram commonly contains processors, devices, connections. Nodes are displayed in browsers window. A processor is a software component capable of executing programs. A device is a hardware component with no computingpower. A connection represents some type of hardware coupling between two entities.



**Fig 6.13: Deployment Diagram for Voice SMS**

**Description:** In this deployement diagram, we have a device called mobile device and processors are mobile storage, user, and Google API.these provide a model to client server system.

Chapter 7

**IMPLEMENTATION** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**7.1 Voice SMS ALGORITHM**

**Input** : User speaks their voice message

**Output**  : TTS will conver voice to speech and sends it to the desired user as both voice

And text message.

The following is the step by step procedure of the implementation of the application:

**Step 1**: Copy the “Voice SMS” application i.e. .apk file in android device.

**Step 2**: Install this app on mobile and open it.

**Step 3**: Home screen will be opened .Now drawable list will be opened. The list contains the following phases

* Contact no
* Message body
* Note making
* Reminder making

**Step 4**: After cliking on the contact no. the user have to press mike buton and speak the message and have to press the send button.

**Step 5**: If invalid, an alert message is dispalyed, stating as message sending is failed. If the contact no. field is empty.

**Step 6**: If valid,then a notification will be displayed as message is sent.

**Step 7**: When any new message is received then user has to turn on the data connection and press the “activate message speaking” button then the received message will be spoken out as voice.if the user wants to stop the message the they can press the same button which acts as “deactivate message speaking”.

**Step 8**: The app also contains extra features such as note making and remainder setting.If user wants to create any note then they have to click on the title of the note and create title,below of it contains another field as Note content field in which user can input their content. After entering the fields then we have to press the Set note button.For making reminder also same it follows the same procedure but it contains the fields as day and notification fields.the user has to input the Day of the eminder and the desired conten to be set as reminder and press the “set reminder” button.

**Step 9**: If any previously enter data is there then it will be displayed along with the newly enter data.

**7.2 PSEUDO CODE**

**/\*Send Message\*/**

protected void sendSMSMessage() {

Log.i("Send SMS", "");

String phoneNo = etContacts.getText().toString();

String message = etMessage.getText().toString();

try {

SmsManager smsManager = SmsManager.getDefault();

smsManager.sendTextMessage(phoneNo, null, message, null, null);

Toast.makeText(getApplicationContext(), "SMS sent.", Toast.LENGTH\_LONG).show();

}

catch (Exception e) {

Toast.makeText(getApplicationContext(), "SMS faild, please try again.", Toast.LENGTH\_LONG).show();

e.printStackTrace();

}

}

**/\*Getting SMS Sending Capability From Mobile\*/**

private void initializeSMSReceiver(){

smsReceiver = new BroadcastReceiver(){

@Override

public void onReceive(Context context, Intent intent) {

Bundle bundle = intent.getExtras();

if(bundle!=null){

Object[] pdus = (Object[])bundle.get("pdus");

for(int i=0;i<pdus.length;i++){

byte[] pdu = (byte[])pdus[i];

SmsMessage message = SmsMessage.createFromPdu(pdu);

String text = message.getDisplayMessageBody();

String sender = getContactName(message.getOriginatingAddress());

speaker.pause(SHORT\_DURATION);

speaker.speak("You have a new message from" + sender + "!");

speaker.pause(SHORT\_DURATION);

speaker.speak(text);

smsSender.setText("Message from " + sender);

smsText.setText(text);

}

}

}

};

}

**/\* Reading the Message\*/**

private void registerSMSReceiver() {

IntentFilter intentFilter = new IntentFilter("android.provider.Telephony.SMS\_RECEIVED");

registerReceiver(smsReceiver, intentFilter);

}

**/\*Setting Read Flag to Message\*/**

protected void gotoRead() {

finish();

Intent i = new Intent(this, ReadActivity.class);

i.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(i);

}

**/\*Disabling TTS\*/**

public void destroy(){

tts.shutdown();

}

**/\*XML fields forto Make Note\*/**

<string name="title1">MAKING NOTE</string>

<string name="title2">SET REMINDER</string>

<string name="title3">Title</string>

<string name="title4">Day</string>

<string name="title5">Note</string>

<string name="title6">Notification</string>

<TableRow

android:id="@+id/tableRow4"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="0.3">

<TextView

android:id="@+id/textView5"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="145dp"

android:text="@string/title5" />

<TextView

android:id="@+id/textView6"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/title6" />

</TableRow>

**/\*Reading Note Fields\*/**

final EditText edTitle = (EditText) findViewById(R.id.etTitle);

final EditText edNote = (EditText) findViewById(R.id.etNote);

**/\*Storing the Notes into Text File\*/**

FileOutputStream openedFile = openFileOutput("supernotes", MODE\_APPEND);

writer = new BufferedWriter(new OutputStreamWriter(openedFile));

String eol = System.getProperty("line.separator");

writer.append(title + eol);

writer.append(note + eol);

writer.close();

Chapter 8

**TESTING**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**8.1 TESTING OBJECTIVES**

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

* Testing is a process of executing a program with the intent of finding an error.
* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable standards.

**8.2 LEVELS OF TESTING**

**8.2.1 Unit Testing:**

Unit testing focuses verification efforts on the smallest unit of the software design**,** the module**.**Unit testing is a method by which individual units of [source code](http://en.wikipedia.org/wiki/Source_code), sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In [procedural programming](http://en.wikipedia.org/wiki/Procedural_programming), a unit could be an entire module, but it is more commonly an individual function or procedure. In[object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming), a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by [white box testers](http://en.wikipedia.org/wiki/White-box_testing) during the development process.

**8.2.2 Integration Testing:**

The objective is to take unit tested modules and build a program structure**.** All the modules are combined and tested as a whole**.**Integration testing is the phase in [software testing](http://en.wikipedia.org/wiki/Software_testing) in which individual software modules are combined and tested as a group. It occurs after [unit testing](http://en.wikipedia.org/wiki/Unit_testing)and before [validation testing](http://en.wikipedia.org/wiki/Verification_and_validation_(software)). Integration testing takes as its input [modules](http://en.wikipedia.org/wiki/Module_(programming)) that have been [unit tested](http://en.wikipedia.org/wiki/Unit_testing), groups them in larger aggregates, applies tests defined in an integration [test plan](http://en.wikipedia.org/wiki/Test_plan)to those aggregates, and delivers as its output the integrated system ready for [system testing](http://en.wikipedia.org/wiki/System_testing).

**6.2.3 System Testing:**

System testing is the stage of implementation that is aimed at ensuring that the system works accurately and efficiently for live operation commences. System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified[requirements](http://en.wikipedia.org/wiki/Requirements). System testing falls within the scope of [black box testing](http://en.wikipedia.org/wiki/Black_box_testing), and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed [integration testing](http://en.wikipedia.org/wiki/Integration_testing) and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

**8.3 BLACK BOX TESTING**

Black-box testing is a method of software testing that tests the functionality of an application as opposed to its internal structures or workings. Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is only aware of what the software is supposed to do, but not how i.e. when he enters a certain input, he gets a certain output; without being aware of how the output was produced in the first place. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure.

This method of test can be applied to all levels of software testing: unit, integration, system and acceptance.

Typical black-box test design techniques include:

* Decision table testing
* [All-pairs testing](http://en.wikipedia.org/wiki/All-pairs_testing)
* [State transition tables](http://en.wikipedia.org/wiki/State_transition_table)
* [Equivalence partitioning](http://en.wikipedia.org/wiki/Equivalence_partitioning)

|  |
| --- |
| **TestCase1** |

**Fig 8.1: Output Screens for Home Page**

**Test Case for Home Page**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No | Description | Test Data | Expected Result | Actual Result |
| 1. | On Succesfull Installation of Application user sees the above page | The user needs to click on Appication in Application launcher icon | Home Page is Displayed | pass |

**Table 8.1: Test Case for Home page**

|  |  |
| --- | --- |
| **Test Case 1:**contact no. insertion | **Test Case 2:** Messsage spoken |

**Fig 8.2:Output Screens for Test Case for Message Sending.**

**Test Case for Contact no. :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No | Description | Test Data | Expected Result | Actual Result |
| 1. | Click on speak icon appered in the application to insert no. | Mobile number=” ” | If no user found with this mobile number | pass |
| 2. | Click on speak icon appered in the application to insert no. | Mobile number=”987653421” | If user found with this mobile number | Pass |

**Table 8.2: Test Case for Sending SMS**

**Table 8.3: Test Case for Message Confirmation**

|  |  |
| --- | --- |
| **Test case 1:** If validation is successfull. | **Test case 2:** if one of the field is invalid. |

**Fig 8.3: Output Screens for Test Case for Forget Password**

**Test Case for Make Request**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No | Description | Test Data | Excepted Result | Actual Result |
| 1. | By clicking send sms button Button without entering valid credentials | Contact no=”9876543210”  Message Body=”can we meet tommorow” | SMS sent. | Pass |
| 2. | By clicking submit button with entering problem and comment | Contact no=” ”  Message Body=”can we meet tommorow” | SMS failed,please try again | Pass |

**Table 8.4: Test Case for Make Note**

|  |  |
| --- | --- |
| **Test Case 1:** if the problem is not entered | **Test Case 2:** on proper submit of Note |

**Fig 8.4: Output Screens Test Case for Test Case for Make Note**

**Test Case for the make note status**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No | Description | Test Data | Excepted Result | Actual Result |
| 1. | On setting the correct title,body the user intened then use is taken into entire list of Notes | Tile=”grocery”  Note=”food items ,shopping list” | Display theNotes page with previous Notes. | Pass |

**Table 8.5: Test Case for the Message Reading**

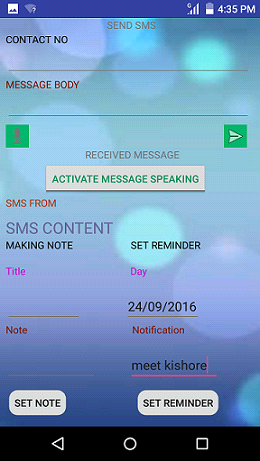
|  |  |
| --- | --- |
| **Test case 1:** if Message reading is not presentare present | **Test case 2:** if message reading is activated |

**Fig 8.5: Output Screens for Test Case for Activating and Deactivating SMS Reading**

**Test Case for Doctor Login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No | Description | Test Data | Excepted Result | Actual Result |
| 1. | Click on the toggle button to activate message speaking | - | tollge button text chenged as expected | Pass |
| 2. | Click on Toggle button to deactivate message speaking | - | toggle button text changed as expected | Pass |

**Table 8.6: Test Case for Reminder**



**Fig 8.6: Output Screens for Test Case for setting reminder**

**Test Case for Reminder**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test no | Description | Test Data | Excepted Result | Actual Result |
| 1. | Click on APP Icon | - | Display the view of the Application Home | Pass |
| 2. | Click on Day to set the reminder | Day=  ”24/09/2016” | Display the Input message | Pass |
| 3. | Click on Notification area | Notification=”meet kishore” | Display the reminder | Pass |

**Table 8.7: Test Case for Reminder**

Chapter 9

**CONCLUSION AND FUTURE SCOPE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**9.1 CONCLUSION**

The application software helps the users to easy acess their message content through speech format. The application is developed for android based Operating System employed in mobile phones. This application is developed to read ,send text messages and voice messages easily.

The software is user-friendly without involving many operations and no computers or dedicated software are required. It is an open source, freely accessible software to benefit the user whoever need this app. Installation of the app in the smartphone is quite simple and more useful to the physically diabled people.people as well as normal people usage.

The goals that are achieved by the Application are:

* Instant access.
* Easily undersatandable
* Clear speech voice
* Reusability
* Simplification of the operations.
* Less processing time and getting required information
* Flexible for further enhancement.

**9.2 FUTURE SCOPE**

It is not possible to develop a system that meets all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

* As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
* Because it is based on object-oriented design, any further changes can be easily adaptable.
* The software need to incorporate more security and privacy, scope for payment of fees to the doctor directly to the specified bank account and more physiological data for chronic diseases
* The software need to incorporate more security and privacy.

Chapter 10

**BIBILOGRAPHY**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­­­­­­­­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_­­­­­­­­­­­­­­­­­­**

**10.1 BOOKS**

1. Dave Thau, The Book of JavaScript 2nd Edition. San Francisco: William Pollock, 2007.

2 .Dynamic HTML: The Definitive Reference by Danny Goodman: California: O’Reily, 2003.

3. Web Technologies: A Computer Science Perspective by Jackson Pearson Education, 2007.

4. Software Testing Techniques by Boris Beizer, Dreamtech Press, 2002.

5. The Unified Modeling Language User Guide, 2/E by Booch, Pearson Education, 2005.

**10.2 WEB SITES**

6.Android Developer Guide: [*http://developer.android.com/guide/index.html*](http://developer.android.com/guide/index.html)*.*

7.Android API: [*http://developer.android.com/reference/packages.html*](http://developer.android.com/reference/packages.html)

8.Java 6 API: [*http://download-llnw.oracle.com/javase/6/docs/api/*](http://download-llnw.oracle.com/javase/6/docs/api/)

9.Google Maps API: [*http://code.google.com/android/add-ons/google-apis/reference/com/google/android/maps/package-summary.html*](http://code.google.com/android/add-ons/google-apis/reference/com/google/android/maps/package-summary.html)

10.Android Fundamentals: [*http://developer.android.com/guide/topics/fundamentals.html*](http://developer.android.com/guide/topics/fundamentals.html)

11.The Java Tutorials: [*http://download-llnw.oracle.com/javase/tutorial/index.html*](http://download-llnw.oracle.com/javase/tutorial/index.html)

12.Android Native Development Kit: [*http://developer.android.com/sdk/ndk/index.html*](http://developer.android.com/sdk/ndk/index.html)

13.Android User Interfaces: [*http://developer.android.com/guide/topics/ui/index.html*](http://developer.android.com/guide/topics/ui/index.html)

Chapter 11

**APPENDIX**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**11.1 OUTPUT SCREENS**

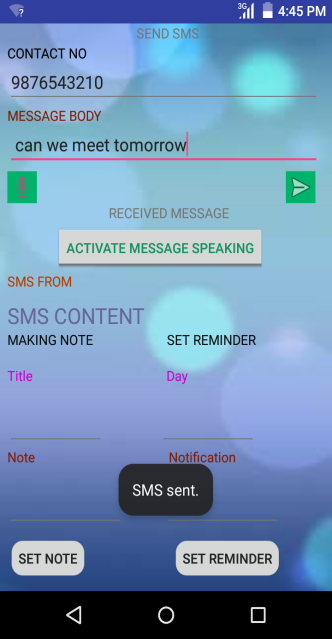


Fig 8.7: Output Screens for successful message sending

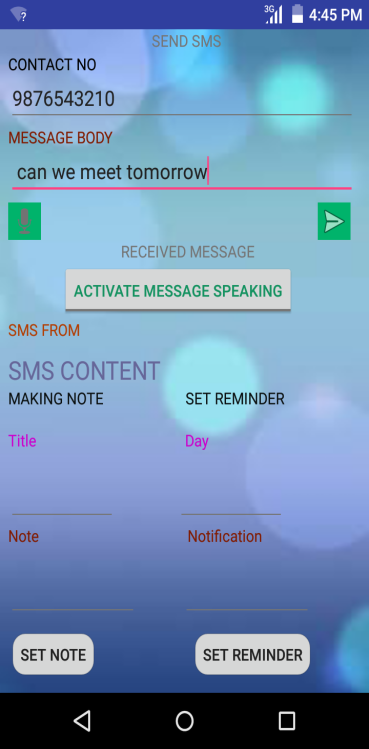
****

Fig 8.8: Output Screens for inserting message into body

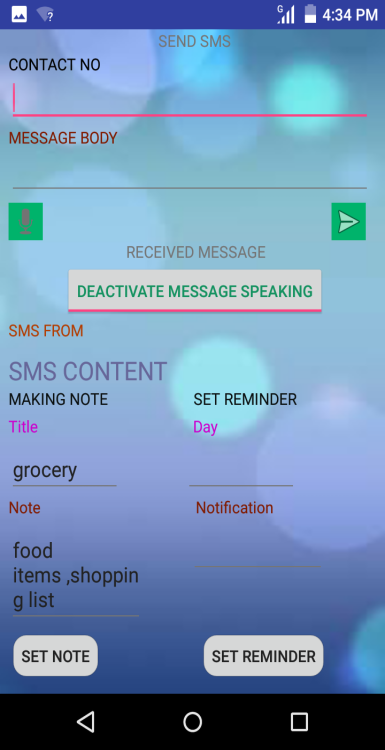


Fig 8.9: Output Screens for Test home page

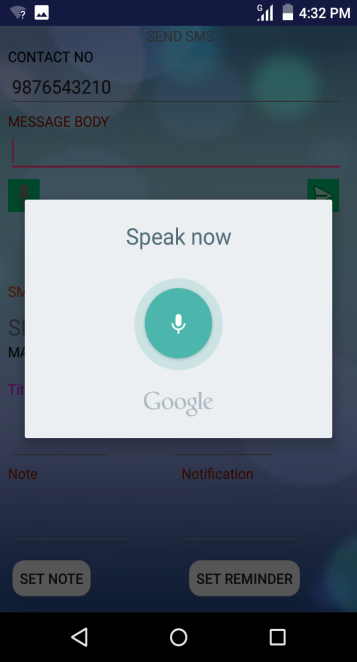
****

Fig 8.10: Output Screens for speaking message

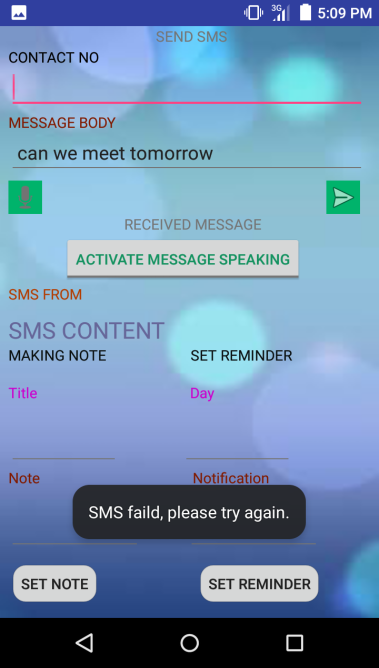


Fig 8.11: Output Screens for message sending failed



Fig 8.12: Output Screens for viewing notes

**11.2 SOURCE CODE**

**MainActivity.java**

package com.example.vsn.voicesms;

import android.app.Activity;

import android.content.ActivityNotFoundException;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;

import android.database.Cursor;

import android.net.Uri;

import android.os.Bundle;

import android.provider.ContactsContract.PhoneLookup;

import android.support.v4.view.accessibility.AccessibilityNodeInfoCompat;

import android.telephony.SmsManager;

import android.telephony.SmsMessage;

import android.util.Log;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.CompoundButton;

import android.widget.CompoundButton.OnCheckedChangeListener;

import android.widget.EditText;

import android.widget.ImageButton;

import android.widget.TextView;

import android.widget.Toast;

import android.widget.ToggleButton;

import java.io.BufferedWriter;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.util.ArrayList;

public class MainActivity extends Activity {

protected static final int RESULT\_SPEECH = 1;

private final int CHECK\_CODE;

private final int LONG\_DURATION;

private final int SHORT\_DURATION;

private ImageButton btnSend;

private ImageButton btnSpeak;

private TextView etContacts;

private TextView etMessage;

private BroadcastReceiver smsReceiver;

private TextView smsSender;

private TextView smsText;

private Speaker speaker;

private ToggleButton toggle;

private OnCheckedChangeListener toggleListener;

private TextView txtText;

/\* renamed from: com.example.vsn.voicesms.MainActivity.1 \*/

class C02121 implements OnClickListener {

C02121() {

}

public void onClick(View view) {

MainActivity.this.sendSMSMessage();

}

}

/\* renamed from: com.example.vsn.voicesms.MainActivity.2 \*/

class C02132 implements OnCheckedChangeListener {

C02132() {

}

public void onCheckedChanged(CompoundButton view, isChecked) {

if (isChecked) {

MainActivity.this.speaker.allow(true);

MainActivity.this.speaker.speak("Okay! Message Reading is Activated");

return;

}

MainActivity.this.speaker.speak("Okay! Message Reading is Deactivated.");

MainActivity.this.speaker.allow(false);

}

}

/\* renamed from: com.example.vsn.voicesms.MainActivity.3 \*/

class C02143 implements OnClickListener {

C02143() {

}

public void onClick(View v) {

Intent intent = new Intent("android.speech.action.RECOGNIZE\_SPEECH");

intent.putExtra("android.speech.extra.LANGUAGE\_MODEL", "en-US");

try {

MainActivity.this.startActivityForResult(intent, MainActivity.RESULT\_SPEECH);

} catch (ActivityNotFoundException e) {

Toast.makeText(MainActivity.this.getApplicationContext(), "Opps! Your device doesn't support Speech to Text", 0).show();

}

}

}

/\* renamed from: com.example.vsn.voicesms.MainActivity.4 \*/

class C02154 implements OnClickListener {

final /\* synthetic \*/ EditText val$edNote;

final /\* synthetic \*/ EditText val$edTitle;

C02154(EditText editText, EditText editText2) {

this.val$edTitle = editText;

this.val$edNote = editText2;

}

public void onClick(View v) {

FileNotFoundException e;

IOException e2;

String title = this.val$edTitle.getText().toString();

String note = this.val$edNote.getText().toString();

Log.d("supernotes", "title is --> " + title);

Log.d("supernotes", "note is --> " + note);

try {

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(MainActivity.this.openFileOutput("supernotes", AccessibilityNodeInfoCompat.ACTION\_PASTE)));

BufferedWriter bufferedWriter;

try {

String eol = System.getProperty("line.separator");

writer.append(title + eol);

writer.append(note + eol);

writer.close();

MainActivity.this.gotoRead();

bufferedWriter = writer;

} catch (FileNotFoundException e3) {

e = e3;

bufferedWriter = writer;

e.printStackTrace();

} catch (IOException e4) {

e2 = e4;

bufferedWriter = writer;

e2.printStackTrace();

}

} catch (FileNotFoundException e5) {

e = e5;

e.printStackTrace();

} catch (IOException e6) {

e2 = e6;

e2.printStackTrace();

}

}

}

/\* renamed from: com.example.vsn.voicesms.MainActivity.5 \*/

class C02165 extends BroadcastReceiver {

C02165() {

}

public void onReceive(Context context, Intent intent) {

Bundle bundle = intent.getExtras();

if (bundle != null) {

Object[] pdus = (Object[]) bundle.get("pdus");

for (int i = 0; i < pdus.length; i += MainActivity.RESULT\_SPEECH) {

SmsMessage message = SmsMessage.createFromPdu((byte[]) pdus[i]);

String text = message.getDisplayMessageBody();

String sender = MainActivity.this.getContactName (message.getOriginatingAddress());

MainActivity.this.speaker.pause (1200);

MainActivity.this.speaker.speak ("You have a new message from" + sender + "!");

MainActivity.this.speaker.pause(1200);

MainActivity.this.speaker.speak(text);

MainActivity.this.smsSender.setText("Message from " + sender);

MainActivity.this.smsText.setText(text);

}

}

}

}

public MainActivity() {

this.CHECK\_CODE = RESULT\_SPEECH;

this.LONG\_DURATION = 5000;

this.SHORT\_DURATION = 1200;

}

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(C0217R.layout.activity\_main);

this.etMessage = (TextView) findViewById(C0217R.id.etMessage);

this.etContacts = (TextView) findViewById(C0217R.id.etContacts);

this.btnSend = (ImageButton) findViewById(C0217R.id.btnSend);

this.btnSpeak = (ImageButton) findViewById(C0217R.id.btnSpeak);

this.toggle = (ToggleButton) findViewById(C0217R.id.speechToggle);

this.smsText = (TextView) findViewById(C0217R.id.sms\_text);

this.smsSender = (TextView) findViewById(C0217R.id.sms\_sender);

EditText edTitle = (EditText) findViewById(C0217R.id.etTitle);

EditText edNote = (EditText) findViewById(C0217R.id.etNote);

Button btnSave = (Button) findViewById(C0217R.id.btnSave);

this.btnSend.setOnClickListener(new C02121());

this.toggleListener = new C02132();

this.toggle.setOnCheckedChangeListener(this.toggleListener);

checkTTS();

initializeSMSReceiver();

registerSMSReceiver();

this.btnSpeak.setOnClickListener(new C02143());

btnSave.setOnClickListener(new C02154(edTitle, edNote));

}

private void checkTTS() {

Intent check = new Intent();

check.setAction("android.speech.tts.engine.CHECK\_TTS\_DATA");

startActivityForResult(check, RESULT\_SPEECH);

}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

switch (requestCode) {

case RESULT\_SPEECH /\*1\*/:

if (resultCode == -1 && data != null) {

ArrayList<String> text = data.getStringArrayListExtra("android.speech.extra.RESULTS");

if (!this.etMessage.isFocused()) {

if (this.etContacts.isFocused()) {

this.etContacts.setText((CharSequence) text.get(RESULT\_SPEECH));

break;

}

}

this.etMessage.setText((CharSequence) text.get(RESULT\_SPEECH));

break;

}

break;

}

if (requestCode != RESULT\_SPEECH) {

return;

}

if (resultCode == RESULT\_SPEECH) {

this.speaker = new Speaker(this);

return;

}

Intent install = new Intent();

install.setAction("android.speech.tts.engine.INSTALL\_TTS\_DATA");

startActivity(install);

}

protected void sendSMSMessage() {

Log.i("Send SMS", BuildConfig.FLAVOR);

try {

SmsManager.getDefault().sendTextMessage(this.etContacts.getText().toString(), null, this.etMessage.getText().toString(), null, null);

Toast.makeText(getApplicationContext(), "SMS sent.", RESULT\_SPEECH).show();

} catch (Exception e) {

Toast.makeText(getApplicationContext(), "SMS faild, please try again.", RESULT\_SPEECH).show();

e.printStackTrace();

}

}

private void initializeSMSReceiver() {

this.smsReceiver = new C02165();

}

private void registerSMSReceiver() {

registerReceiver(this.smsReceiver, new IntentFilter("android.provider.Telephony.SMS\_RECEIVED"));

}

private String getContactName(String phone) {

Uri uri = Uri.withAppendedPath(PhoneLookup.CONTENT\_FILTER\_URI, Uri.encode(phone));

String[] projection = new String[RESULT\_SPEECH];

projection[0] = "display\_name";

Cursor cursor = getContentResolver().query(uri, projection, null, null, null);

if (cursor.moveToFirst()) {

return cursor.getString(0);

}

return "unknown number";

}

protected void onDestroy() {

super.onDestroy();

unregisterReceiver(this.smsReceiver);

this.speaker.destroy();

}

protected void gotoRead() {

finish();

Intent i = new Intent(this, ReadActivity.class);

i.addFlags(268435456);

startActivity(i);

}

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(C0217R.menu.menu\_main, menu);

return true;

}

}

**Read Activity.java**

package com.example.vsn.voicesms;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.widget.TextView;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStreamReader;

public class ReadActivity extends Activity {

protected void onCreate(Bundle savedInstanceState) {

FileNotFoundException e;

IOException e2;

super.onCreate(savedInstanceState);

setContentView(C0217R.layout.read);

try {

BufferedReader reader = new BufferedReader(new InputStreamReader(openFileInput("supernotes")));

BufferedReader bufferedReader;

try {

String eol = System.getProperty("line.separator");

StringBuilder sb = new StringBuilder();

TextView tvNotes = (TextView) findViewById(C0217R.id.notes);

while (true) {

String line = reader.readLine();

if (line != null) {

sb.append(line).append(eol);

} else {

tvNotes.setText(sb.toString());

bufferedReader = reader;

return;

}

}

} catch (FileNotFoundException e3) {

e = e3;

bufferedReader = reader;

e.printStackTrace();

} catch (IOException e4) {

e2 = e4;

bufferedReader = reader;

e2.printStackTrace();

}

} catch (FileNotFoundException e5) {

e = e5;

e.printStackTrace();

} catch (IOException e6) {

e2 = e6;

e2.printStackTrace();

}

}

public void onBackPressed() {

super.onBackPressed();

finish();

Intent i = new Intent(this, MainActivity.class);

i.addFlags(268435456);

startActivity(i);

}

}

**Speaker Activity.java**

package com.example.vsn.voicesms;

import android.content.Context;

import android.speech.tts.TextToSpeech;

import android.speech.tts.TextToSpeech.OnInitListener;

import java.util.HashMap;

import java.util.Locale;

public class Speaker implements OnInitListener {

private boolean allowed;

private boolean ready;

private TextToSpeech tts;

public Speaker(Context context) {

this.ready = false;

this.allowed = false;

this.tts = new TextToSpeech(context, this);

}

public boolean isAllowed() {

return this.allowed;

}

public void allow(boolean allowed) {

this.allowed = allowed;

}

public void onInit(int status) {

if (status == 0) {

this.tts.setLanguage(Locale.US);

this.ready = true;

return;

}

this.ready = false;

}

public void speak(String text) {

if (this.ready && this.allowed) {

HashMap<String, String> hash = new HashMap();

hash.put("streamType", String.valueOf(5));

this.tts.speak(text, 1, hash);

}

}

public void pause(int duration) {

this.tts.playSilence((long) duration, 1, null);

}

public void destroy() {

this.tts.shutdown();

}

}

**SOURCE CODE OF XML FILES**

main\_menu.xml

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

tools:context="com.example.vsn.voicesms.MainActivity">

<item

android:id="@+id/action\_settings"

android:orderInCategory="100"

android:title="@string/action\_settings"

app:showAsAction="withText" />

</menu>

activity\_main.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:paddingBottom="@dimen/padding\_small"

android:paddingLeft="@dimen/padding\_small"

android:paddingRight="@dimen/padding\_small"

android:paddingTop="@dimen/padding\_small"

android:weightSum="1.0"

android:background="#6644"

tools:context=".MainActivity"

android:paddingStart="1dp">

<TextView

android:id="@+id/sms\_no"

android:layout\_width="wrap\_content"

android:layout\_height="@dimen/padding\_large"

android:layout\_below="@+id/sms\_sender"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="1dp"

android:text="@string/sms\_to"

android:textAppearance="?android:attr/textAppearanceListItemSmall"

android:textColor="#124960" />

<EditText

android:id="@+id/etContacts"

android:gravity="top"

android:inputType="textMultiLine"

android:lines="1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/etSearchHint" />

<TextView

android:id="@+id/sms\_body"

android:layout\_width="wrap\_content"

android:layout\_height="@dimen/padding\_large"

android:layout\_below="@+id/sms\_sender"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="1dp"

android:text="@string/sms\_body"

android:textAppearance="?android:attr/textAppearanceListItemSmall"

android:textColor="#604912" />

<EditText

android:id="@+id/etMessage"

android:gravity="top"

android:inputType="textMultiLine"

android:lines="1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

/>

<TableLayout

android:id="@+id/tableLayout1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" >

<TableRow

android:id="@+id/tableRow1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="0.3">

<ImageButton

android:id="@+id/btnSpeak"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

tools:context=".MainActivity"

android:layout\_marginRight="255dp"

android:layout\_marginTop="4dp"

android:src="@android:drawable/ic\_btn\_speak\_now" />

<ImageButton

android:id="@+id/btnSend"

tools:context=".MainActivity"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginRight="4dp"

android:layout\_marginTop="4dp"

android:text="@string/send"

android:src="@android:drawable/ic\_menu\_send" />

</TableRow>

</TableLayout>

<ToggleButton

android:id="@+id/speechToggle"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:layout\_marginLeft="70dip"

android:textOff="@string/speech\_toggle\_on"

android:textOn="@string/speech\_toggle\_off"

android:textColor="#149260"

/>

<TextView

android:id="@+id/sms\_sender"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true"

android:text="@string/sms\_label"

android:textColor="#ff0040"

android:textAppearance="?android:attr/textAppearanceSmall" />

<TextView

android:id="@+id/sms\_text"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/sms\_sender"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="10dp"

android:text="@string/sms\_message"

android:textAppearance="?android:attr/textAppearanceLarge"

android:textColor="#4000ff" />

<TableRow

android:id="@+id/tableRow2"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="0.3">

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="#121212"

android:layout\_marginRight="80dp"

android:text="@string/title1" />

<TextView

android:id="@+id/textView2"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textColor="#121212"

android:text="@string/title2" />

</TableRow>

<TableRow

android:id="@+id/tableRow3"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="0.3">

<TextView

android:id="@+id/textView3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="145dp"

android:text="@string/title3" />

<TextView

android:id="@+id/textView4"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/title4" />

</TableRow>

<TableRow

android:layout\_height="wrap\_content"

android:layout\_width="match\_parent">

<EditText

android:id="@+id/etTitle"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="60dp"

android:ems="5">

</EditText>

<EditText

android:id="@+id/etDay"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:ems="5">

</EditText>

</TableRow>

<TableRow

android:id="@+id/tableRow4"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="0.3">

<TextView

android:id="@+id/textView5"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="145dp"

android:text="@string/title5" />

<TextView

android:id="@+id/textView6"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/title6" />

</TableRow>

<TableRow

android:id="@+id/tableRow5"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="0.3">

<EditText

android:id="@+id/etNote"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="44dp"

android:ems="6">

</EditText>

<EditText

android:id="@+id/etNotify"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:ems="6">

</EditText>

</TableRow>

<TableRow

android:id="@+id/tableRow6"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="0.3">

<Button

android:id="@+id/btnSave"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginRight="80dp"

android:text="@string/save1" />

<Button

android:id="@+id/btnSave1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/save2" />

</TableRow>

</LinearLayout>

read.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical" android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:id="@+id/notes"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>

strings.xml

<resources>

<string name="app\_name">Voice Sms</string>

<string name="action\_settings">Settings</string>

<string name="btSpeak">Speak</string>

<string name="tvTextMatches">Text Matches</string>

<string name="sNoOfMatches"> No of Matches</string>

<string name="etSearchHint"></string>

<string name="sms\_to">CONTACT NO</string>

<string name="sms\_body">MESSAGE BODY</string>

<string name="sms\_label">SMS FROM</string>

<string name="send">SEND</string>

<string name="sms\_message">SMS CONTENT</string>

<string name="speech\_toggle\_on">ACTIVATE MESSAGE SPEAKING</string>

<string name="speech\_toggle\_off">DEACTIVATE MESSAGE SPEAKING</string>

<string name="start\_speaking">Okay! Message Reading is Activated.</string>

<string name="stop\_speaking">Okay! Message Reading is Deactivated.</string>

<string name="title1">MAKING NOTE</string>

<string name="title2">SET REMINDER</string>

<string name="title3">Title</string>

<string name="title4">Day</string>

<string name="title5">Note</string>

<string name="title6">Notification</string>

<string name="save1">SET NOTE</string>

<string name="save2">SET REMINDER</string>

</resources>

androidmanifest.xml

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.vsn.voicesms" >

<uses-permission android:name="android.permission.INTERNET"/>

<uses-permission android:name="android.permission.READ\_SMS"/>

<uses-permission android:name="android.permission.RECEIVE\_SMS"/>

<uses-permission android:name="android.permission.READ\_CONTACTS"/>

<uses-permission android:name="android.permission.SEND\_SMS" />

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name=".MainActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity

android:name="com.example.vsn.voicesms.ReadActivity"

android:label="@string/app\_name" >

</activity> </application>