**SHORT MESSAGE VOICING USING SPEECH SYNTHESIS ENGINE**

**A PROJECT REPORT**

***Submitted by***

**MOHAMMAD ALI KHAN A - 21108205132**

**SRINIVASAN N - 21108205091**

**VIJAY V - 21108205102**

***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

INFORMATION TECHNOLOGY

**RAJALAKSHMI ENGINEERING COLLEGE**

**ANNA UNIVERSITY: CHENNAI 600 025**

APRIL 2012

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**SHORT MESSAGE VOICING USING SPEECH SYNTHESIS ENGINE**” is the bonafide work of “**MOHAMMAD ALI KHAN A,** **SRINIVASAN N, VIJAY V**” who carried out the project under my supervision.

**SIGNATURE SIGNATURE**

Mr. P. KUMAR, M.E Ms. S .BHAGAVATHIPRIYA,

**HEAD OF THE DEPARTMENT** M. Tech

Information Technology  **SUPERVISOR**

Rajalakshmi Engineering College, Associate Professor

Thandalam, Chennai – 602105 Information Technology

Rajalakshmi Engineering College,

Thandalam, Chennai – 602105

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We thank the Almighty God for the successful completion of the project. Sincere thanks to our Chairman **Mr. S. Meganathan B.E, F.I.E** for his sincere endeavour in educating us in his premier institution.

We would like to express our deep gratitude to our beloved chair person **Dr. (Mrs).Thangam Meganathan, Ph.D.,** for her enthusiastic motivation which inspired us a lot in completing this project.

We also express our gratitude to our principal, **Dr. S. Renganarayanan, B.E., M.Sc., Ph.D.,** who helped us in providing the required facilities in completing the project.

We would like to thank our Head of the Department **Mr. P. Kumar, M.E.,** for his invaluable guidance and encouragement for completing the project.

We would also like to thank our supervisor **Mrs. S. Baghavathi Priya, M.Tech** for constantly guiding us and motivating us throughout the course of the project.

We are very thankful to our project coordinator **Mrs. L. Priya, M.Tech** for her invaluable guidance, ideas, advice and encouragement for the successful completion of this project.

We express our gratitude to our parents and friends for extending their full support to us.

i

**ABSTRACT**

The progress of mobile communication with recent advances, especially in the text to speech technology results in potential opportunities for message-voicing applications. Techniques that have been previously functional for text to speech have limitations for SMS voicing. The focus is to develop an android application, which could translate text message (SMS) to speech. Users can access the message without reading it. With text as input, a voice of that text can be heard as output. Thereby the device is enabled to speak the text. Welfare of this is to help the disable, help during driving. The expenses of call cost can also be reduced with voice output. A Text-to-Speech synthesis engine (TTS) is used, which converts the text to speech. The voice output is stored in .WAV format in the mobile SD card. A device such as a music player can be used to play the saved voice when needed. Along with this Messaging and quick Mailing facilities are provided. Voice recognition is also supported by the application. Input to SMS or mail can be either text or voice. This makes the application hands free where message can be heard and replied through voice.

ii

**LIST OF FIGURES**

**FIGURE NO. NAME PAGE NO.**

4.1 Usecase diagram

4.2 Receive text DFD

4.3 Voice output DFD

4.4 System architecture

4.5 Application icon

4.6 Application screen

6.1 Project explorer

6.2 Run Android applications

6.3 Emulator

6.4 Application runs in emulator

6.5 Creation of test project

6.6 Choose project to test

6.7 Result of JUnit test

6.8 Result of Android Lint test

6.9 Shell command

6.10 Toast displaying the entire message

6.11 .WAV format files in SD card

6.12 Export application

6.13 Creating keystore

6.14 Generation of keystore and .apk files

6.15 About the application

6.16 Application fan page

6.17 Toast displaying the SMS

6.18 Options for received SMS

6.19 .WAV file creation

6.20 Voice recognition

6.21 SMS and Mail interface

**LIST OF NOTATIONS**

**ENTITY**

**USECASE**

**DATA STORE**

**PROCESS**

**UNIDIRECTIONAL FLOW**

**ACTOR**

**LIST OF ABBREVATIONS**

**S.NO ABBREVATIONS PAGE NO.**

1SMV – Short Message Voicing

2TTS – Text to Speech

3PC – Personal Computer

4 GPS – Global Positioning System

5 JDK – Java Development Kit

6 SDK – Software Development Kit

7 DVM – Davlik Virtual Machine

8 DEX – Dalvik Executable Files

9 GUI – Graphical User Interface

10 SSL – Secure Socket Layer

11 IMAP – Internet Mail Access Protocol

12 APK – Application Package

13 AVD – Android Virtual Device

14 GPRS – General Packet Switching Network

15 SD – Storage Device

16 XML – Extensible Mark-up Language

17 SMS – Short Message Service

18 WAV – Waveform

19 IDE – Integrated development Environment

20 ADT – Android Development Toolkit

21 MMS – Multimedia Messaging Service

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**ACKNOWLEDGEMENT** i

**ABSTRACT**

**LIST OF FIGURES**

**LIST OF NOTATIONS**

**LIST OF ABBREVATIONS**

**1.0 INTRODUCTION**

1.1 Purpose of the project

1.2 Merits

1.3 Limitations

**2.0 LITERATURE SURVEY**

2.1 Natural Language Processing

With Text-to-Speech on Android

2.2 Speech Activated Telephony E-mail

Reader (SATER) Based on Speaker

Verification and Text-to-Speech

Conversion

2.3 Proposed System

**3.0 PROBLEM FORMULATION**

3.1 Objective

3.2 Methodology

3.3 Platform Requirements

3.3.1 Hardware Requirements

3.3.2 Software Requirements

**4.0 SYSTEM ANALYSIS AND DESIGN**

4.1 Usecase diagram

4.2 Data Flow diagram

4.3 Architecture diagram

4.4 Table design

4.5 Application design

**5.0 FUNCTIONAL DESCRIPTION**

5.1 LIST OF MODULES

5.1.1 Receive text

5.1.2 Voice output

**6.0 APPLICATION DEVELOPMENT, TESTING AND**

**IMPLEMENTAION**

6.1 Receive text

6.2 Voice output

6.3 Unit testing

6.4 Android Lint

6.5 Functional testing

6.5.1 UI/Application exerciser

6.6 Test cases

6.6.1 Character support up to 765

6.6.2 Instant voice output to SMS

6.7 Implementation

**7.0 CONCLUSION AND FUTURE ENHANCEMENT**

7.1 Conclusion

7.2 Future enhancement

**8.0 BIBLIOGRAPHY**

8.1 Books

8.2 Websites

**CHAPTER – 1**

**INTRODUCTION**

**1.1 Purpose of the project**

The SMV application using TTS engine is used to convert the incoming text message to voice output. This application aims at providing a user friendly application so that users can access the message without seeing it. The message is converted in the form such that the mobile will speak out the message and the contact name from whom the message was received.

This system has the capability to speak out the person name from whom it is received. This is applicable only if the sender phone number is saved in the phone contact, if not the application will read out the phone number when message is received. Using this application the uses can send a reply to the message.

In this application the users can configure the Email id. Once the email is configured users can send mail from this application itself. The mail sent and received from this email id will be stored in the different label when the same configure account is viewed through a browser in PC. This application will be useful for users while driving. For instance, users can know the message received without accessing it.

**1.2 Merits**

* Enables the mobile to speak.
* The application provides high user friendly interface.
* The application helps disabled people.
* The application helps users while driving.

**1.3 Limitation**

* The application requires General Packet Radio Service connection to send and receive E-mail.

**CHAPTER – 2**

**LITERATURE SURVEY**

**2.1 Natural Language Processing with Text-to-Speech on Android**

**(May 2011)**

**Sonal Bhatt, Graduate Student, Arizona State University, Division of Computing Studies.**

The paper is about guiding users on Cab availability through Android. This used the TTS technology to return the results of query as a voice. The users can query the details of a cab in a particular locality. GPS system is used to find out the cab details. Internet connection is a necessary one to make this application work. The user query is taken in the form of text whereas the result from the GPS system is returned as a voice. The voice output tells about the details enquired about the cab. However processing of long messages is not possible by the application and the result is in the form of text rather than voice.

**2.2 Speech Activated Telephony E-mail Reader (SATER) Based on**

**Speaker Verification and Text-to-Speech Conversion (August 2009)**

**Chung-Hsien Wu and Jay-Hung Chen**

Speech Activated Telephony E-mail Reader (SATER) is an integrated system combining speaker verification, network, and text-to-speech conversion. A registered user can activate and listen to the Email through a wired/wireless telephone. SATER is an integrated system combining speaker verification, network, and text-to-speech conversion. In this system, the person first enters the claimed identity via speech, and then on request from a verification system utters a computer prompted verification digit string, and requests to listen to his Email or voice mail assuming he is verified. Therefore, users can access E-mail contents at any time and at any place via telephone. However the user is unable to use the system when he is unwell.

**2.3 Proposed System**

**Short Message voicing using Speech Synthesis Engine**

**Mohamed Ali Khan A, Srinivasan N, Vijay V**

The Short Message system using speech synthesis engine (SMV) is used to convert the incoming text message to voice output. This application aims at providing the customers a user friendly application so that customers have access the message without viewing it. The message is converted in the form text in such a way that it will speak out the message and from whom the message is received. This system has the capability to read out the sender’s name. This is possible only if the message sender phone number is saved in the phone contact, if not the engine will read out the phone number from which it received. This application will be useful for customers who are driving. If any customer is driving and any message is received the system will automatically reads the message. If the user is not the situation to hear the message he can stop reading the message by using a button given in the application and later he can hear the last received message. The proposed project overcomes the limitations of all the existing ones.

**CHAPTER – 3**

**PROBLEM FORMULATION**

**3.1 Objective**

To develop an Android application that converts the incoming text message to speech using the TTS engine.

**3.2 Methodology**

The methodology behind the Short Messaging voicing application using speech synthesis engine is that the application uses the Text-to-Speech technology that enables the mobile to speak. The input text message is fetched from the phone memory and it is sent to the speech synthesis engine that converts the text input to voice.

**3.3 Platform requirements**

**3.3.1** Hardware requirements

* Android Mobile (Version 2.3.3 – Gingerbread)
* i559 processor with 800 MHz
* 384 MB RAM

**3.3.2** Software Requirements

* Java Development Kit 7(JDK)
* Android SDK
* Eclipse Indigo 3.7.1
* ADT Plug-in
* User Interface : XML
* Programming Language : Java 1.6

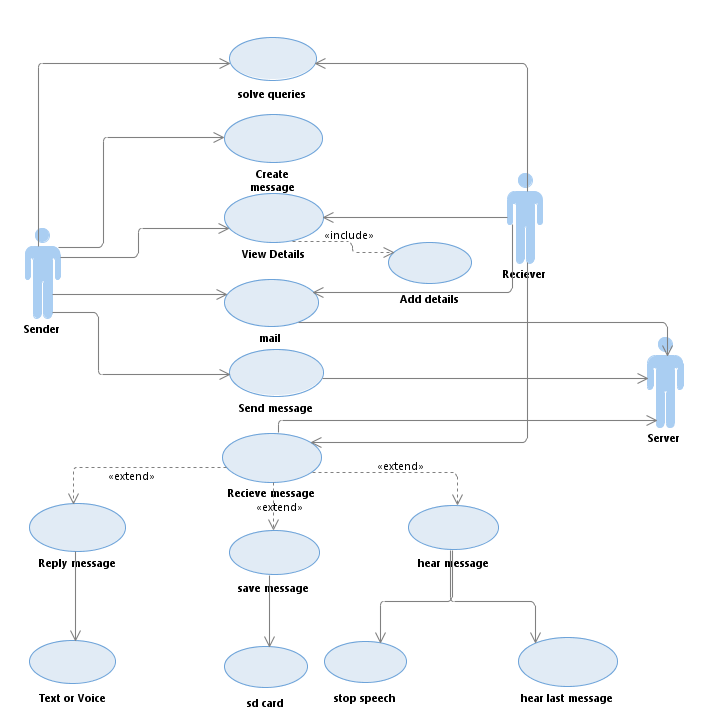
**CHAPTER – 4**

**SYSTEM ANALYSIS AND DESIGN**

System analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning. This involves studying the business processes, gathering operational data, understanding the informational flow, finding out the bottlenecks and evolving the solutions for overcoming the weaknesses of the system so as to achieve the organizational goals.

**4.1 UseCase diagram**

Use case diagram presents a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

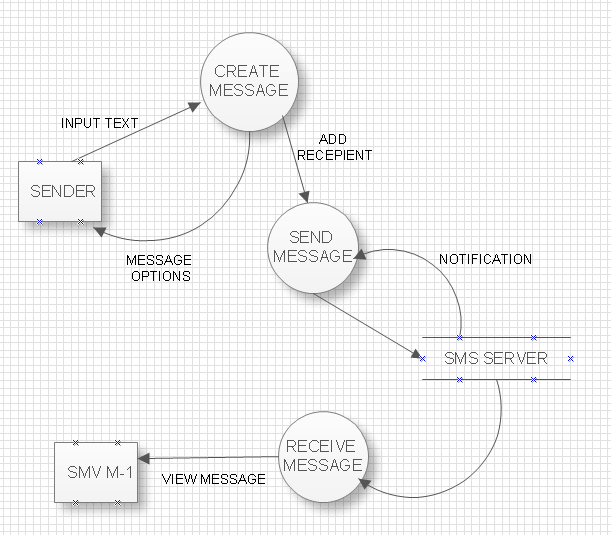


**Figure 4.1 Usecase diagram**

The use case diagram for Short Message Voicing System Using Speech Synthesis Engine is shown in Figure. It involves three types of actors namely Sender, Server and Receiver. User installs the application in his/her android mobile and whenever any message received in user mobile the application takes the text message as input and converts it in to voice output. The user can reply to the received message, the message will be sent through the server and then it will be sent to the receiver. The user can delete the received message. The user can save the message and can hear the last received message.

**4.2 Data-Flow Diagram**

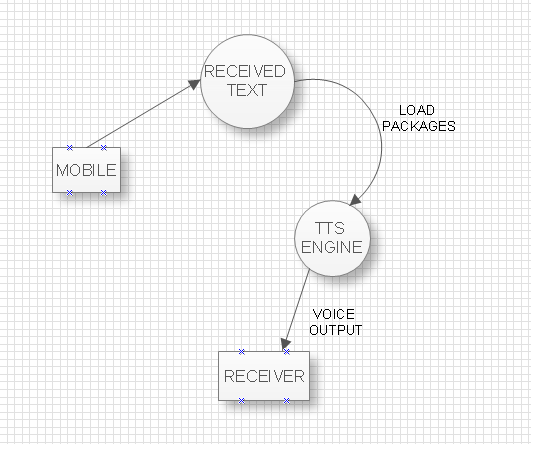
**Receive text**

****

**Figure 4.2 Receive text DFD**

User interacts with mobile by installing the application into it. When sender sends the input text message using create message default option and sends to the server and the server redirects to the receiver where the user installed the application. Now the input text message is fetched from the receivers mobile to the application and now the application is ready to convert the text message to speech.

**Voice output**

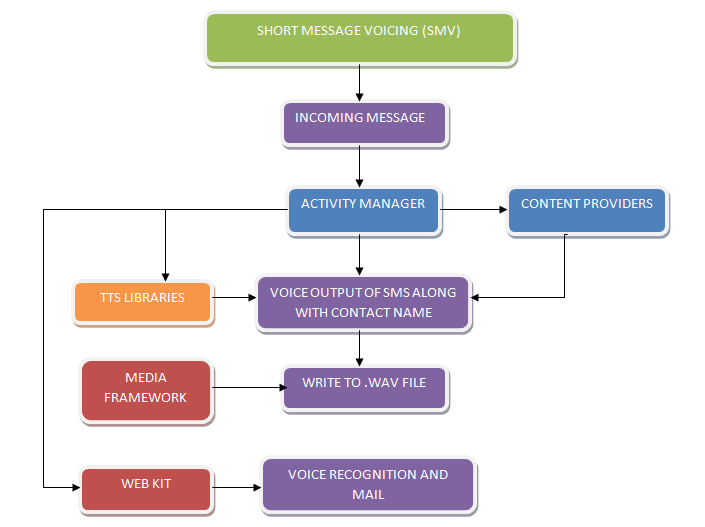
****

**Figure 4.3 Voice output DFD**

The application fetches the text message from the phone memory and it will send it to TTS engine and the voice output is heard out at the receiver end. The user can reply to the message through the application if needed or else delete the message.

**4.3 Architecture diagram**

The system architecture is built on Linux. It is the lower most layer of the architecture which consists of the drivers. Above it is the Libraries built using C or C++ includes the web kit, media framework. Along with it is the DVM. This is responsible for converting the .class java files to .dex files. This type of virtual machine is used for small processors like the mobile. DVM is built on C. Third layer above is the Application framework built in java. The TTS libraries, Activity manager are included here. At the top of all is the application which is developed using java.



**Figure 4.4 System architecture**

**4.4 Table design**

The application speaks the contact name of the person from which the message is received. The contacts in an Android mobile get stored in the SQlite database. Contacts are stored in the file contacts.db from where they can be accessed. SQlite does not allow third party applications to access its contents. It is a secured data store. Content providers are present above the SQlite. Content providers allow third party users or applications to access the contacts table. The SMV application makes use of these content providers by which the application speaks the contact name when the message is received.

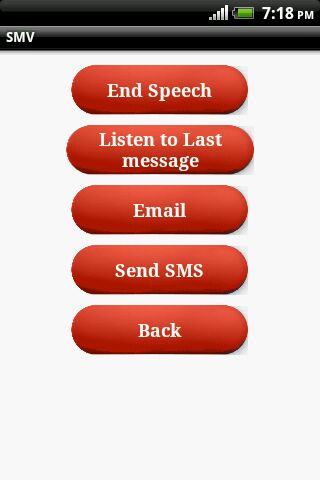
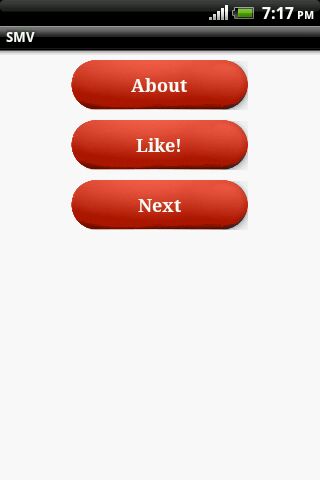
**4.5 Application design**

The GUI is designed using XML. Since Short Message voicing System using Speech Synthesis Engine is developed as an android application, the layouts and buttons are designed using XML. The Icon of the Application along with the page layout for Application Home Screen, E-mail and Send SMS is shown below.



**Figure 4.5 Application icon**

The Icon for the Short Message Voicing Using Speech Synthesis Engine is shown in Figure. It denotes the application in the android mobile.

****

**Figure 4.6 Application screen**

The layout of the application home screen is shown in Figure. The home screen displays the **“About”**, **“Next”** and **“Like ”** commands the on pressing the next button displays the **“End speech”, “Listen to Last Message”, “Email”** and **“Send SMS”** buttons to perform the respective actions. The “**Back**” button helps user to get back to the home screen. The user can choose **“Like!”** button which redirects to Facebook where the user can give feedback about the applications and can know about the updates of the application.

**CHAPTER – 5**

**FUNCTIONAL DESCRIPTION**

**5.1 List of modules**

The modules in the project are

5.1.1 Receive text

5.1.2 Voice output

**5.1.1 Receive text**

The users of the system include the sender and the receiver. The SMS server also is a part of this system. The user inputs the text message with the available options and sends the text message. It reaches the SMS server from which the message is forwarded to the directed number mentioned by the sender. Android mobile by default receives the message and stores it in the inbox. This module is designed to work such a way that the message received on the other end can be accessed by the application developed. Through this the message is accessed without opening the mobile inbox. The application receives the message and stores it. The next step is to provide a voice output to the text message received.

**5.1.2 Voice output**

The received text message is available in the application developed. TTS libraries are included to the text message on the receiver side. TTS enables the conversion of text into voice output of several languages. Few of which include English in US, UK accent, Italian, Japanese, Spanish, etc. Using this engine the text is converted to voice output in English UK accent language. The user on the receiver side can hear the voice output of the text message. As a result the mobile is enabled to speak the text. With the result of this module the text received by the application developed automatically speaks out the message. The voice output of the message also gets stored in mobile SD card in .WAV format. Media player present by default in Android mobile can be used to play the audio file.

**CHAPTER – 6**

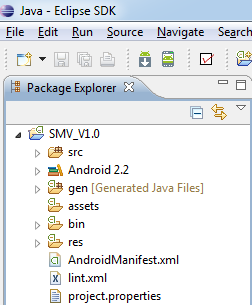
**APPLICATION DEVELOPMENT, TESTING AND IMPLEMENTATION**

**6.1 Receive text**

Application development defines a sequence of steps or modules in which each module use the result of the previous module. Implementation is the stage of the project when the theoretical design is turned into a working system. If the implementation stage is not carefully planned and controlled, it can cause chaos. Thus it can be considered to be the most crucial stage in achieving a successful new application and in giving the users confidence that the new application will work and will be effective and user friendly.

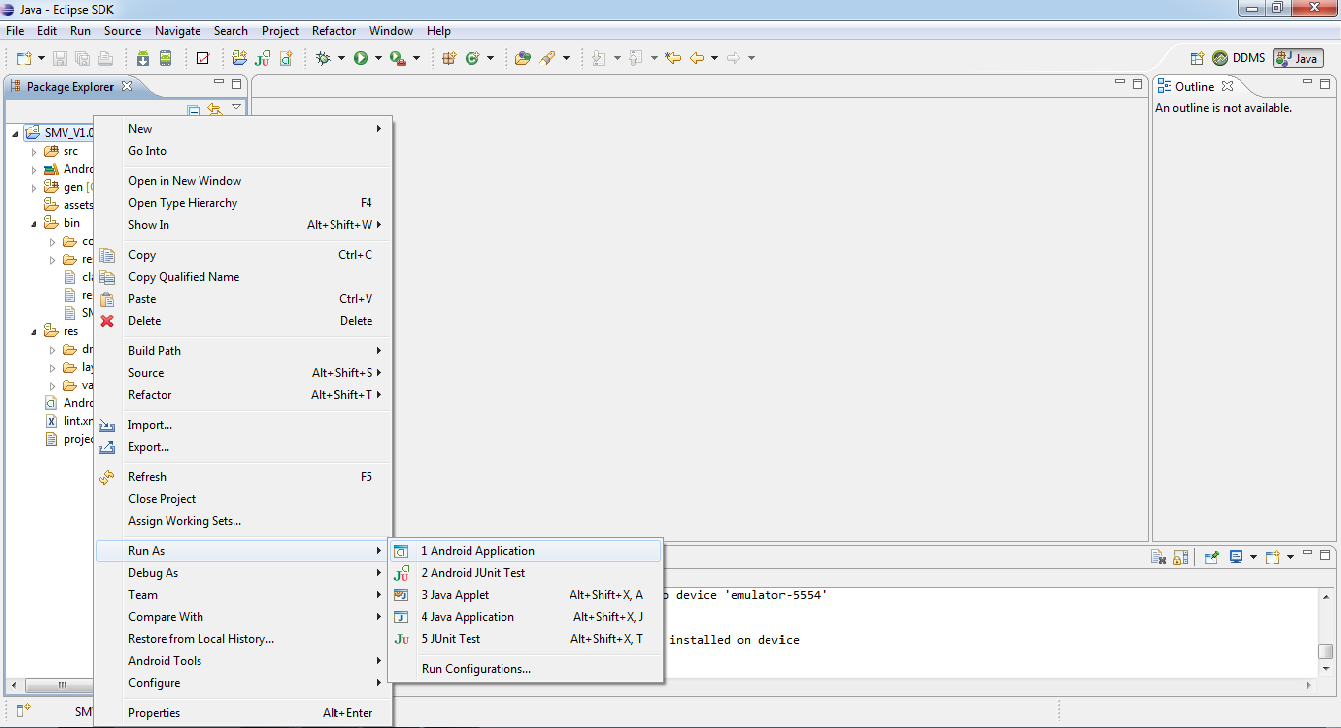
The first task involved in the implementation stage of Short Message Voicing Using Speech Synthesis Engine is planning that is deciding on the modules and the methods to be adopted for developing the application. Once the planning has been completed, the major effort is to develop the application and ensure that the functionalities in the applications are working properly. Planning, development, Debugging and testing are considered as the preparatory activities of implementation.

Eclipse is used for developing applications. It integrates with the SDK; ADT is the plug-in which is used to create a powerful IDE which is used to build Android applications. AVD and hardware devices are used for testing. Using eclipse the Android SDK, Emulator, AVD can be worked with. A new project is created with the desired name. The created project appears as shown in the Figure.



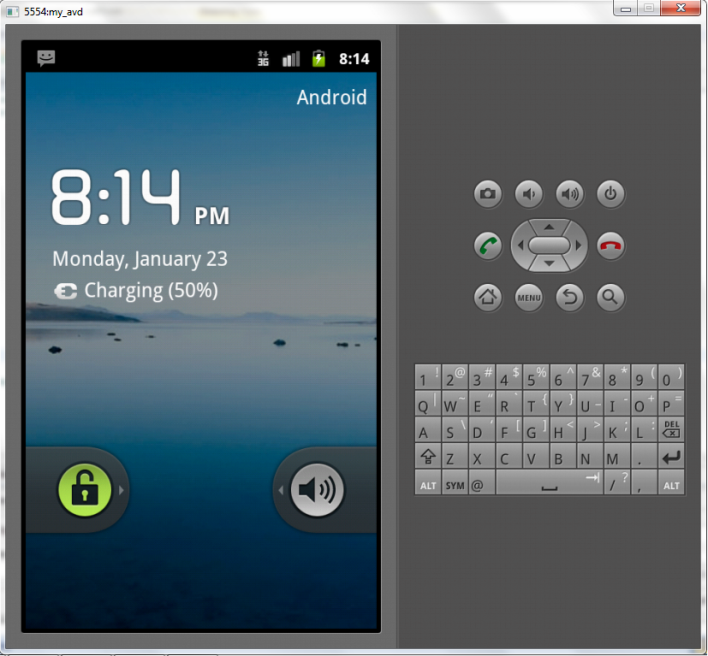
**Figure 6.1 Project explorer**

The src contains all the source files, the coding is written here and saved as java files. The gen includes the automatically generated java files. Android 2.3.3 is the version of the platform used for the Android application development. It includes all the packages, classes needed for development. AndroidManifest.xml is a file which is used to create the UI and provide permissions for process like RECEIVE\_SMS, SEND\_SMS, and READ\_CONTACTS. res is the resources folder consists of the images and files which are needed for the applications. In order to run the projects right click on the project and select Run As -> Android application and using Android Virtual Device i.e. Emulator it is shown in the Figure.



**Figure 6.2 Run Android applications**

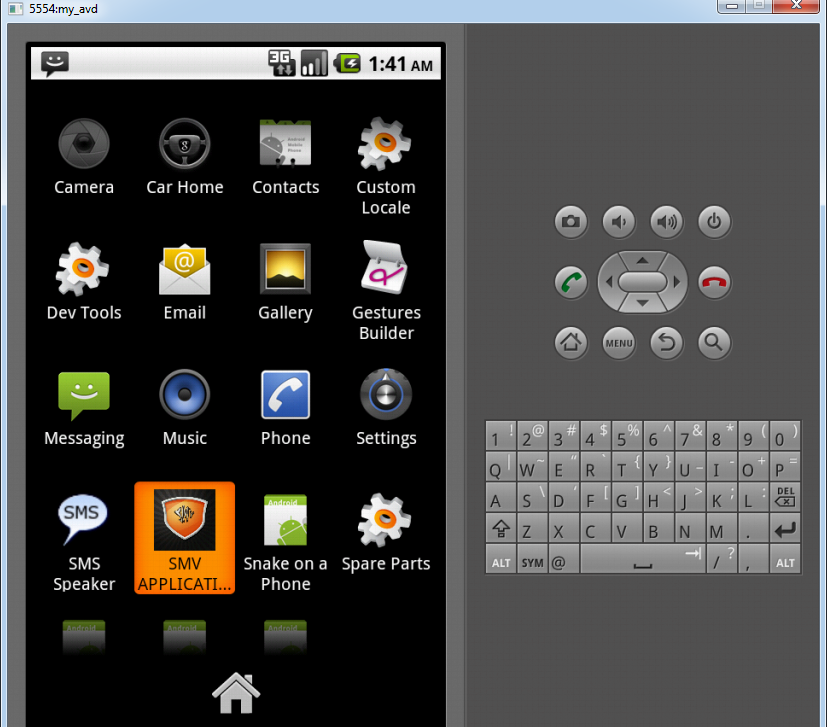
Now the Android Virtual Device i.e. Emulator starts working and it is shown in Figure. The Emulator acts similar to a mobile. It ships within the Android SDK it. All processes carried out in a mobile can also be done in the emulator. Other process such as internet services, SD card services can be carried out.



**Figure 6.3 Emulator**

**6.2 Voice output**

Now the application gets installed in the emulator. Using the emulator the application functionalities can be tested and the installed application looks like as shown in the Figure.



**Figure 6.4 Application runs in emulator**

During each of the development process of application, it has been tested using emulator for further enhancements. Emulator can be used to hear the voice output of the message. Also the .WAV file stored in SD card can be played using the media player. At the end of the development process the final result of the application will be in .apk format that gets generated while running the application and this should be copied for the implementation of the in Android mobile.

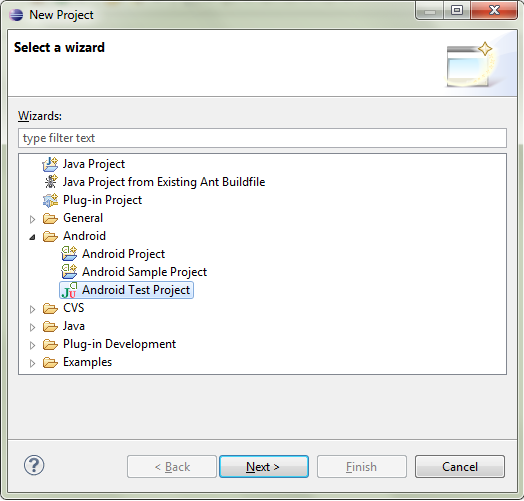
**6.3 Unit testing**

Unit testing is a method by which individual units of source code are tested to determine if they accept valid inputs and reject the invalid ones. A unit may be an individual function, procedure or method. Android components are tested using JUnit. JUnit provides component specific test cases classes.

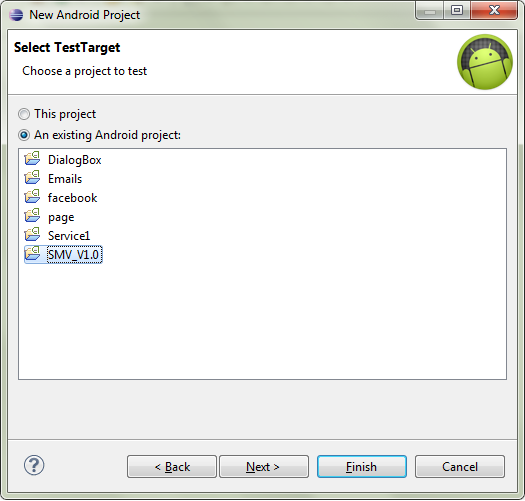
A test package is created similar to the application package. The classes are imported to it and the application is tested. JUnit is an assertion based testing where any one false returned makes the test case to be considered failure. Following is the procedure to carry out JUnit test for the application developed.

With the application developed present in the Eclipse a new Android Test project is created with the same Android platform used for development. Name the test project and set the target as the application developed. The next step is to define a test case class file which has a super class ‘AcivityInstrumentationTestCase’. Import the class file which is to be tested in this java file. Declare a constructor which is used by Android testing framework when we run the test. A setup( ) is added which is called while running the test. Preconditions are included which is the test condition for the application. Finally a unit test method is given to test each method, classes in the coding. Following illustrates graphically how an application is tested using JUnit testing Framework.

An Android test project is created.

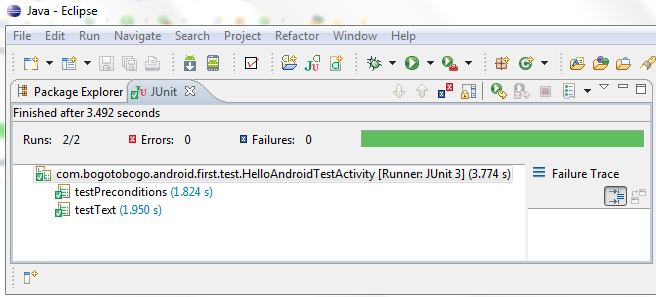


**Figure 6.5 Creation of test project**



**Figure 6.6 Choose project to test**

The test cases created are used to test a particular package or project as a whole. This is done by right clicking the project and selecting run Android JUnit test. The test results are shown in the explorer. The JUnit explorer includes several fields. It displays the time in seconds taken for the test to get over. A progress bar showing the progress indicating the amount completed. Numbers of classes which are run to test are shown. Along with it the Errors and Failures are indicated. Failure trace is present which would display the error details in case of test failure.

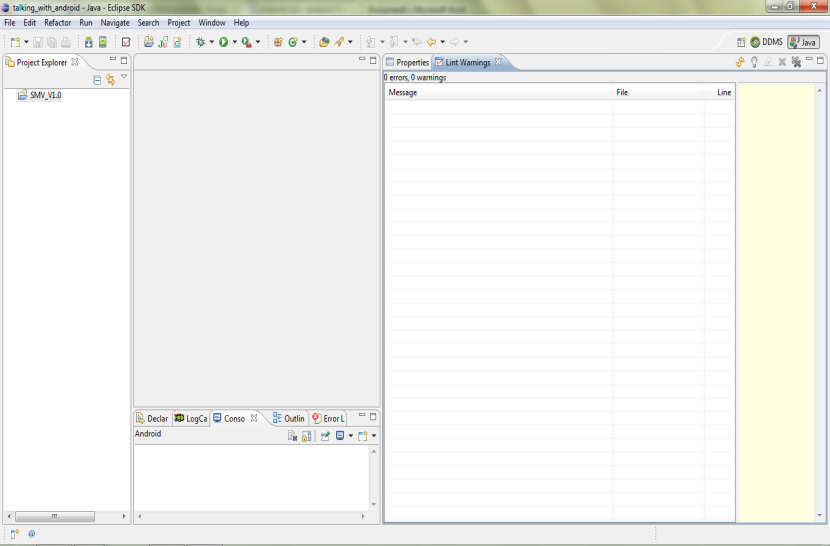


**Figure 6.7 Result of JUnit test**

**6.4 Android Lint test**

Android application testing in Eclipse includes the Android Lint. This is present along with ADT 16 and above. Lint will scan Android project sources for potential bugs. This test could detect inconsistent array size, Manifest errors, unused resources and many more. Lint test is carried out by the following way. Right click the application developed in Eclipse. Choose Run Lint from Android Tools.

A new explorer named Lint Warning is opened. It displays the errors and warnings after checking the application. Option to fix the bug is also available. The result of Lint test for the application was done and it showed no errors or no warnings.



**Figure 6.8 Result of Android Lint test**

**6.5 Functional testing**

Android provides three ways by which Functional testing can be done for Android applications. They are the Monkey Runner, UI/Application Exerciser, Robotium.

Monkey Runner is based on Python Language. Applications which are integrated with Python to carry out functionality are tested using this. Modules for testing are first created in Python and they are run to test the application.

Robotium is an open source third party testing method for Android applications. The tool can be downloaded and the application is imported inside it and tested. Results similar to JUnit testing are obtained. This is generally not an encouraged practise.

**6.5.1 UI/Application exerciser**

This is the most common Functional testing carried out to test the Android applications. This is also referred to as Monkey Test or Stress tester. Random clicks are generated to the application to test its capability. This works along with the Android Debug Bridge. From command line this test can be performed where events are generated to packages which are set as target. This identifies unresponsiveness, looks for crashes and much more. For instance the following command line is used to generate events to the application to stress test it.

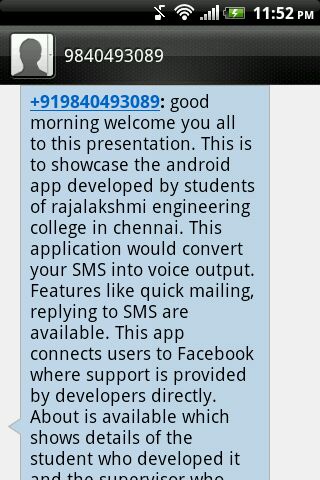
$ adb shell monkey -p your.package.name -v 500

**Figure 6.9 Shell command**

**6.6 Test cases**

**6.6.1 Character support upto 765**

The application receives the message sent by the user on the sender side. On the receiver side it displays the message received. SMV provides support as normal 160 characters for a message. It also produces a voice output of the message. When tested with different conditions the application was found to support voice output for messages upto 765 characters. Above which the SMS gets converted to MMS and it is not detected by the application.



**Figure 6.10 Toast displaying the entire message**

**6.6.2 Instatnt Voice output to SMS**

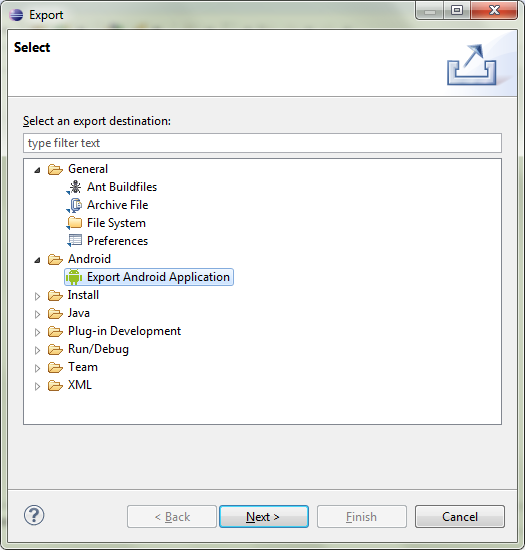
The application enables the Android mobile to speak out the text SMS received. This by default happens when a message is received. Activity starts and voice output can be heard.However when the application was tested it was found to work properly when it received a number of messages. The test was carried out by sending 10 messages at the same time to the Android mobile. The result of the test was that the application started working by which voice output was heard for all messages received one at a time also the .WAV format of the text was also created and stored in the mobile SD card successfully.



**Figure 6.11 .WAV format files in SD card**

**6.7 Implementation:**

The tested application to work the developers should digitally sign it. This process is done using Eclipse itself. Right click the project developed and choose export.



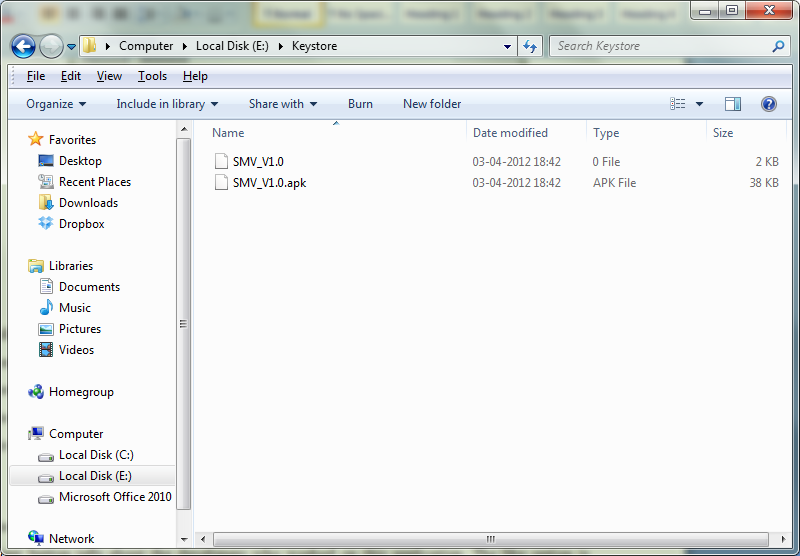
**Figure 6.12 Export Application**

Select the project to export from the projects available in the list. The next window would prompt the developer to create a keystore. The key store should be of a new one; the path where it is to be created is also mentioned.



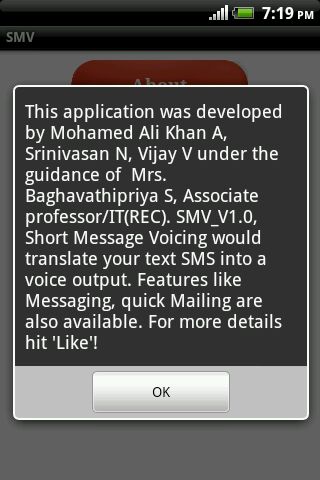
**Figure 6.13 Creating Keystore**

Following the creation the next window would prompt for details such as year of expiry of license, a minimum of 25 years can be given. Organization name, locality, contact are also essential for this. Finally the location where the apk file is to be generated is also mentioned. The folder mentioned includes the keystore along with the .apk file which is the executable file to be installed in Android mobile devices and used.

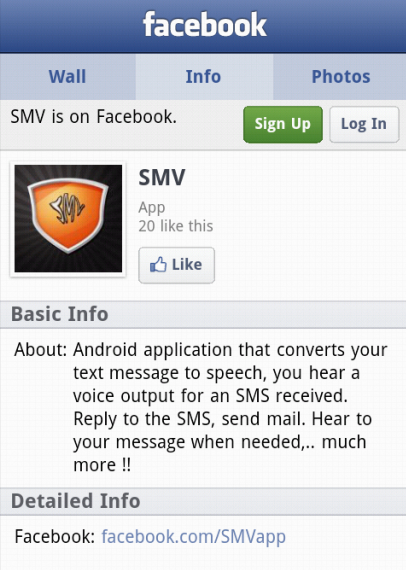


**Figure 6.14 Generation of keystore and .apk file**

The application working includes the following. The home screen displays three options. The about button tells about the developers who worked on this application. The like option is similar to a customer support. Users of the application can come in direct contact to the developers through this. The update on this application or any others improvements are mentioned over here. This makes customer relationship good and queries can be solved easily. This requires internet connectivity for action to perform. Social networking based interaction to customers is introduced through this application.

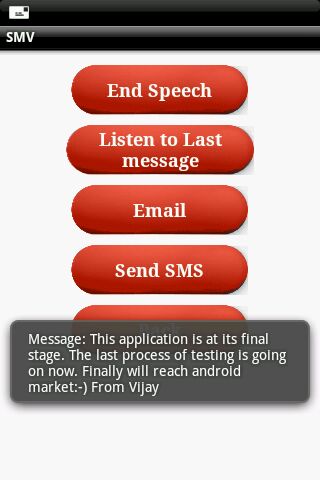


**Figure 6.15 about the application**

****

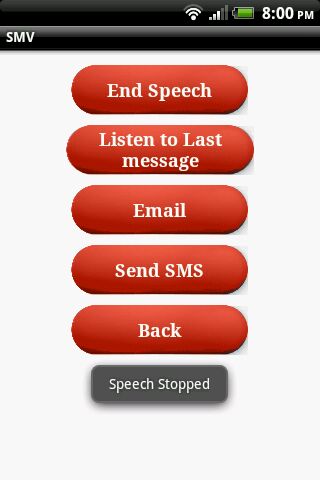
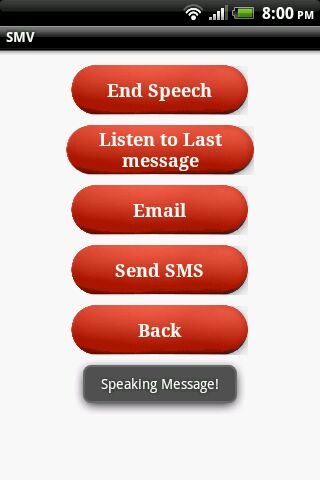
**Figure 6.16 Application fan page**

The next option navigates to the next page of the SMV application. This would welcome the users with a voice saying, ‘welcome to short message voicing application’. Facilities such as mail, SMS are available here. When a message is received the application displays it along with the contact name.



**Figure 6.17 Toast displaying the SMS**

Users if busy and if not interested to hear to the voice they can end it using the end speech option. For instance if the user is ready to hear later at some stage he can listen to the last received message using the listen to last message option. The application speaks again the last received SMS.

**** 

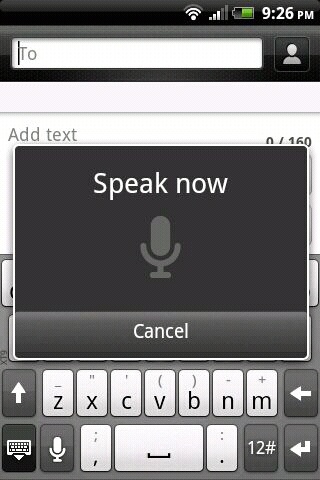
**Figure 6.18 Options for SMS received**

As the mobile speaks out the SMS received the application also does another function. It writes the voice output of the text received in .WAV format. This audio file can be played using media player present by default in Android mobile. The file gets stored in the mobile SD card. The name of the file is starting few characters of the text received.

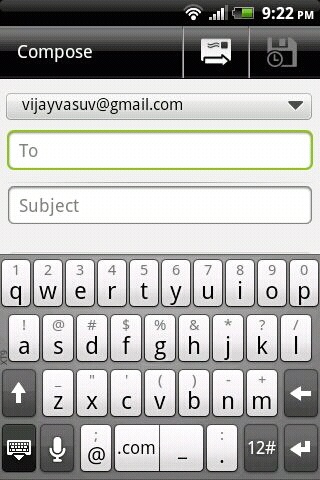
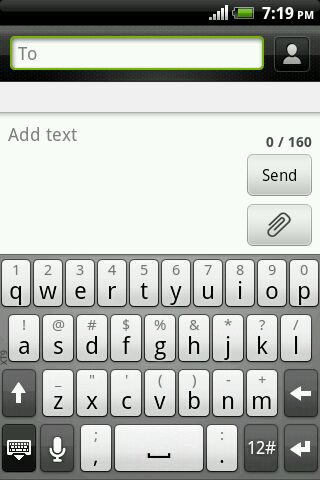
**Figure 6.19 .WAV file creation**

Voice recognition is also applicable through this application. SMS and quick Mailing facility are available. For this the input can either text or voice. This also requires internet connection to work. When connected to net the user speaks the desired message it gets displayed in the mobile using voice recognition. This would make the application completely hands free. The voice output and voice input to the message is possible.



**Figure 6.20 Voice recognition**

This makes the application to provide input to SMS ans mailing. The mail option works once configure with a particular mail id. Users need not login everytime to access the mail. Quick mailing is achieved. Reply to the message received can also be done. Separate interface for it is provided.

**Figure 6.21 SMS and Mail interface**

**CHAPTER – 7**

**CONCLUSION AND FUTURE ENHANCEMENT**

**7.1 Conclusion**

The recent advancements in mobile communication have led to the development of one million applications and more. Those have made the life of people easy in the way they work.

With mobile devices playing a vital role in everyday’s life it is essential to make the application user friendly. The popularity of the applications depends on the way they are developed and handled. It is also dependent on the fact how users are given support by the developers.

In order to satisfy all the above facts we developed an Android Application which makes it user- friendly and ensure customer support 24x7. The application ensures speech output of the received SMS. The user can either opt for text reply or voice reply to the received SMS. Mailing facility is made available in the application which is one of the most important means of communication now-a-days. This is used as a quick mailing function where user need not log in every time.

SMV is the only application that integrates SMS, voice output and Mailing facilities under one roof. Customer support is also provided through social networking sites. User gets updates regarding the application frequently. Queries regarding the application are solved directly by the developers.

**7.2 Future Enhancement**

The future work for this application would make it user friendly for people in rural areas. For any application to survive in the market it need to be updated regularly or enhancements must be made.

The enhancement for our application would be as follow

* Voice output of the Email received.
* Speak out the received SMS and Email in Tamil language.

**CHAPTER – 8**

**BIBLIOGRAPHY**

**8.1 BOOKS**

* Designing Effective Speech Application, JavaTM Speech API Programmer's Guide, Sun Microsystems, Inc.
* Android in Action, 2nd edition, W Frank Ableson, Robi Sen, Chris King
* Beginning Android 3, Apress publications, Mark Murphy
* Beginning Android Application Development Wrox programming, Wei- Meng Lee
* Professional Android Application Development Wrox programming, Reto Meier

**8.2 WEBSITES**

* [http://www.developer.android.com](http://www.developer.android.com/)
* [http://www.market.android.com](http://www.market.android.com/)
* <http://www.code.google.com/android>
* <http://www.flashxml.com/flash-tutorials/resources-for-android-development/>
* <http://mobiforge.com/designing/story/understanding-user-interface-android-part-1-layouts>
* <http://p2p.wrox.com/sms-in-android/>
* <http://thenewboston.org/tutorials/android-message-service>