AN IMPLEMENTATION OF SPEECH RECOGNITION FOR DESKTOP APPLICATION

BY

Name	**Roll No.**
1. **Jibran Abbasi(Group Leader)**	**06SW94**
2. **Muzamil Hussain**	**06SW35**
3. **Shoaib Ahmed**	**06SW70**

Supervised by
(Lecturer Isma Farah)
Department of Software Engineering

Submitted in partial fulfillment of the requirement for the degree
of Engineering in Software.

2010

DEPARTMENT OF SOFTWARE ENGINEERING

CERTIFICATE

This is to Certify that the work in this Thesis / Project titled as **"AN IMPLEMENTATION OF SPEECH RECOGNITION FOR DESKTOP APPLICATION"** is entirely written, successfully completed and demonstrated by the following students themselves as a partial fulfillment of requirement for the Bachelor's of Engineering in Software Engineering.

Mr. Jibran Abbasi	(06SW94)
Mr. Muzamil Hussain	(06SW35)
Mr. Shoaib Ahmed	(06SW70)

Eng. Isma Farah
Thesis Supervisor

Prof.Tahseen Hafiz
Chairman
Depatment of Software Engineering
Date: ______________________

ACKNOWLEDGMENTS

Thanks to Almighty Allah Who gave us courage and understanding to start and later on finish this work of Thesis/Project. Thanks To Our Supervisor who guided and helped us at the different phases of the work.

DEDICATION

This Thesis/Project work is dedicated to our parents and our teachers, who taught us different roads of life and directed us towards our destinations. To all those friends and companions who helped us while doing this work and made the journey of university career easier.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

PROJECT OVERVIEW

This thesis report considers an overview of speech recognition technology, software development, and its applications. The first section deals with the description of speech recognition process, its applications in different sectors, its flaws and finally the future of technology. Later part of report covers the speech recognition process, and the code for the software and its working. Finally the report concludes at the different potentials uses of the application and further improvements and considerations.

1.1 Project Objective

- ❑ To understand the speech recognition and its fundamentals.
- ❑ Its working and applications in different areas
- ❑ Its implementation as a desktop Application
- ❑ Development for software that can mainly be used for:
    - ❑ Speech Recognition
    - ❑ Speech Generation
    - ❑ Text Editing
    - ❑ Tool for operating Machine through voice.

1.2 Abstract

Speech recognition technology is one from the fast growing engineering technologies. It has a number of applications in different areas and provides potential benefits. Nearly 20% people of the world are suffering from various disabilities; many of them are blind or unable to use their hands effectively. The speech recognition systems in those particular cases provide a significant help to them, so that they can share information with people by operating computer through voice input.

This project is designed and developed keeping that factor into mind, and a little effort is made to achieve this aim. Our project is capable to recognize the speech and convert the input audio into text; it also enables a user to perform operations such as

“save, open, exit” a file by providing voice input. It also helps the user to open different system software such as opening Ms-paint, notepad and calculator.

At the initial level effort is made to provide help for basic operations as discussed above, but the software can further be updated and enhanced in order to cover more operations.

1.3 Project scope

This project has the speech recognizing and speech synthesizing capabilities though it is not a complete replacement of what we call a NOTEPAD but still a good text editor to be used through voice. This software also can open windows based softwares such as Notepad, Ms-paint and more.

CHAPTER 2

LITERATURE REVIEW

2.1 An overview of Speech Recognition

Speech recognition is a technology that able a computer to capture the words spoken by a human with a help of microphone [1] [2]. These words are later on recognized by speech recognizer, and in the end, system outputs the recognized words. The process of speech recognition consists of different steps that will be discussed in the following sections one by one.

An ideal situation in the process of speech recognition is that, a speech recognition engine recognizes all words uttered by a human but, practically the performance of a speech recognition engine depends on number of factors. Vocabularies, multiple users and noisy environment are the major factors that are counted in as the depending factors for a speech recognition engine [3].

2.2 History

The concept of speech recognition started somewhere in 1940s [3], practically the first speech recognition program was appeared in 1952 at the bell labs, that was about recognition of a digit in a noise free environment [4], [5].

- ✓ 1940s and 1950s consider as the foundational period of the speech recognition technology, in this period work was done on the foundational paradigms of the speech recognition that is automation and information theoretic models [15].
- ✓ In the 1960's we were able to recognize small vocabularies (order of 10-100 words) of isolated words, based on simple acoustic-phonetic properties of speech sounds [3]. The key technologies that were developed during this decade were, filter banks and time normalization methods [15].
- ✓ In 1970s the medium vocabularies (order of 100-1000 words) using simple template-based, pattern recognition methods were recognized.
- ✓ In 1980s large vocabularies (1000-unlimited) were used and speech recognition problems based on statistical, with a large range of networks for handling

language structures were addressed. The key invention of this era were hidden markov model (HMM) and the stochastic language model, which together enabled powerful new methods for handling continuous speech recognition problem efficiently and with high performance [3].

- ✓ In 1990s the key technologies developed during this period were the methods for stochastic language understanding, statistical learning of acoustic and language models, and the methods for implementation of large vocabulary speech understanding systems.
- ✓ After the five decades of research, the speech recognition technology has finally entered marketplace, benefiting the users in variety of ways. The challenge of designing a machine that truly functions like an intelligent human is still a major one going forward.

2.3 Types of speech recognition

Speech recognition systems can be divided into the number of classes based on their ability to recognize that words and list of words they have. A few classes of speech recognition are classified as under:

2.3.1 Isolated Speech

Isolated words usually involve a pause between two utterances; it doesn't mean that it only accepts a single word but instead it requires one utterance at a time [4].

2.3.2 Connected Speech

Connected words or connected speech is similar to isolated speech but allow separate utterances with minimal pause between them.

2.3.3 Continuous speech

Continuous speech allow the user to speak almost naturally, it is also called the computer dictation.

2.3.4 Spontaneous Speech

At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. An ASR system with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, "ums" and "ahs", and even slight stutters.

2.4 Speech Recognition Process

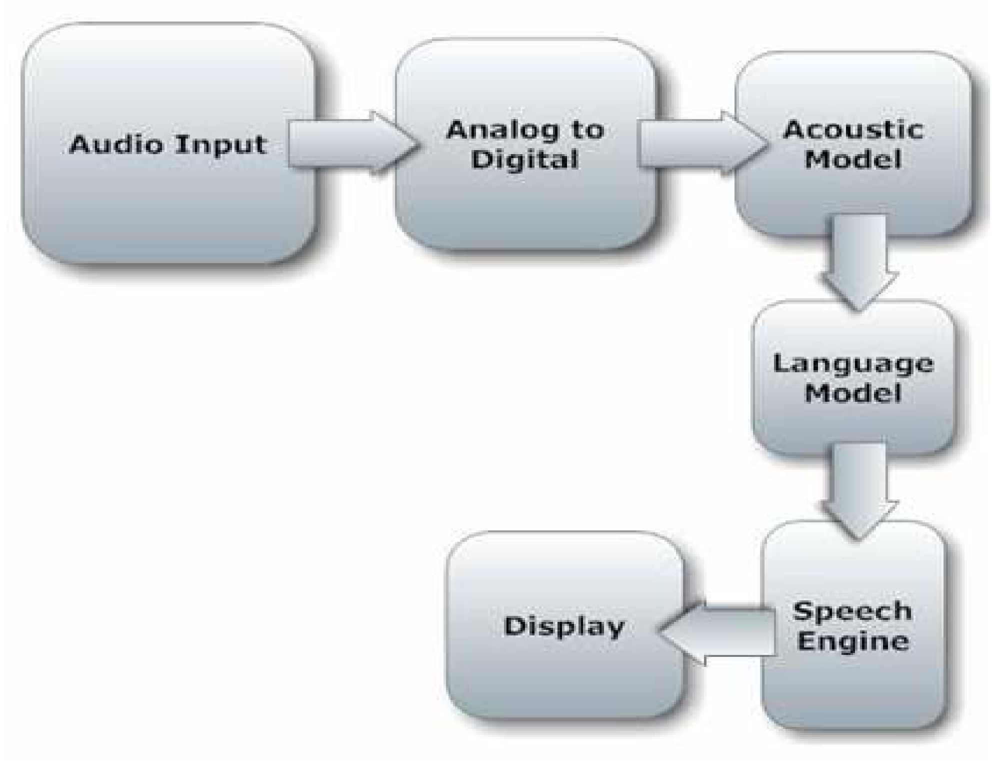


Fig: 2.1 [4] [8] Speech Recognition Process

2.4.1 Components of Speech recognition System

Voice Input

With the help of microphone audio is input to the system, the pc sound card produces the equivalent digital representation of received audio [8] [9] [10].

Digitization

The process of converting the analog signal into a digital form is known as digitization [8], it involves the both sampling and quantization processes. Sampling is converting a continuous signal into discrete signal, while the process of approximating a continuous range of values is known as quantization.

Acoustic Model

An acoustic model is created by taking audio recordings of speech, and their text transcriptions, and using software to create statistical representations of the sounds that make up each word. It is used by a speech recognition engine to recognize speech [8]. The software acoustic model breaks the words into the phonemes [10].

Language Model

Language modeling is used in many natural language processing applications such as speech recognition tries to capture the properties of a language and to predict the next word in the speech sequence [8]. The software language model compares the phonemes to words in its built in dictionary [10].

Speech engine

The job of speech recognition engine is to convert the input audio into text [4]; to accomplish this it uses all sorts of data, software algorithms and statistics. Its first operation is digitization as discussed earlier, that is to convert it into a suitable format for further processing. Once audio signal is in proper format it then searches the best match for it. It does this by considering the words it knows, once the signal is recognized it returns its corresponding text string.

2.5 Uses of Speech Recognition Programs

Basically speech recognition is used for two main purposes. First and foremost dictation that is in the context of speech recognition is translation of spoken words into text, and second controlling the computer, that is to develop such software that probably would be capable enough to authorize a user to operate different application by voice [4][11].

Writing by voice let a person to write 150 words per minute or more if indeed he/she can speak that much quickly. This perspective of speech recognition programs create an easy way for composing text and help the people in that industry to compose millions of words digitally in short time rather then writing them one by one, and this way they can save their time and effort.

Speech recognition is an alternative of keyboard. If you are unable to write or just don’t want to type then programs of speech recognition helps you to do almost any thing that you used to do with keyboard.

2.6 Applications

2.6.1 From medical perspective

People with disabilities can benefit from speech recognition programs. Speech recognition is especially useful for people who have difficulty using their hands, in such cases speech recognition programs are much beneficial and they can use for operating computers. Speech recognition is used in deaf telephony, such as voicemail to text.

2.6.2 From military perspective

Speech recognition programs are important from military perspective; in Air Force speech recognition has definite potential for reducing pilot workload. Beside the Air force such Programs can also be trained to be used in helicopters, battle management and other applications.

2.6.3 From educational perspective

Individuals with learning disabilities who have problems with thought-to-paper communication (essentially they think of an idea but it is processed incorrectly causing it to end up differently on paper) can benefit from the software. Some other application areas of speech recognition technology are described as under [13]:

Command and Control

ASR systems that are designed to perform functions and actions on the system are defined as Command and Control systems. Utterances like "Open Netscape" and "Start a new browser" will do just that.

Telephony

Some Voice Mail systems allow callers to speak commands instead of pressing buttons to send specific tones.

Medical/Disabilities

Many people have difficulty typing due to physical limitations such as repetitive strain injuries (RSI), muscular dystrophy, and many others. For example, people with difficulty hearing could use a system connected to their telephone to convert the caller's speech to text.

2.7 Speech Recognition weakness and flaws

Besides all these advantages and benefits, yet a hundred percent perfect speech recognition system is unable to be developed. There are number of factors that can reduce the accuracy and performance of a speech recognition program.

Speech recognition process is easy for a human but it is a difficult task for a machine, comparing with a human mind speech recognition programs seems less intelligent, this is due to that fact that a human mind is God gifted thing and the capability of thinking, understanding and reacting is natural, while for a computer program it is a complicated task, first it need to understand the spoken words with respect to their meanings, and it has to create a sufficient balance between the words, noise and spaces. A human has a built in capability of filtering the noise from a speech while a machine requires training, computer requires help for separating the speech sound from the other sounds.

Few factors that are considerable in this regard are [10]:

Homonyms: Are the words that are differently spelled and have the different meaning but acquires the same meaning, for example “there” “their” “be” and “bee”. This is a challenge for computer machine to distinguish between such types of phrases that sound alike.

Overlapping speeches: A second challenge in the process, is to understand the speech uttered by different users, current systems have a difficulty to separate simultaneous speeches form multiple users.

Noise factor: the program requires hearing the words uttered by a human distinctly and clearly. Any extra sound can create interference, first you need to place system away from noisy environments and then speak clearly else the machine will confuse and will mix up the words.

2.8 The future of speech recognition.

• Accuracy will become better and better.

• Dictation speech recognition will gradually become accepted.

- Greater use will be made of “intelligent systems” which will attempt to guess what the speaker intended to say, rather than what was actually said, as people often misspeak and make unintentional mistakes.
- Microphone and sound systems will be designed to adapt more quickly to changing background noise levels, different environments, with better recognition of extraneous material to be discarded.

2.9 Few Speech recognition softwares

2.9.1 XVoice

XVoice is a dictation/continuous speech recognizer that can be used with a variety of XWindow applications. This software is primarily for users.

HomePage:http://www.compapp.dcu.ie/~tdoris/Xvoice/
http://www.zachary.com/creemer/xvoice.html
Project: http://xvoice.sourceforge.net

2.9.2 ISIP

The Institute for Signal and Information Processing at Mississippi State University has made its speech recognition engine available. The toolkit includes a front−end, a decoder, and a training module. It's a functional toolkit.

This software is primarily for developers.
The toolkit (and more information about ISIP) is available at: http://www.isip.msstate.edu/project/speech

2.9.3 Ears

Although Ears isn't fully developed, it is a good starting point for programmers wishing to start in ASR. This software is primarily for developers.

FTP site: ftp://svr−ftp.eng.cam.ac.uk/comp.speech/recognition/

2.9.4 CMU Sphinix

Sphinx originally started at CMU and has recently been released as open source. This is a fairly large program that includes a lot of tools and information. It is still "in development", but includes trainers, recognizers, acoustic models, language models, and some limited documentation. This software is primarily for developers.

Homepage: http://www.speech.cs.cmu.edu/sphinx/Sphinx.html
Source: http://download.sourceforge.net/cmusphinx/sphinx2−0.1a.tar.gz

2.9.5 NICO ANN Toolkit

The NICO Artificial Neural Network toolkit is a flexible back propagation neural network toolkit optimized for speech recognition applications.
This software is primarily for developers.
Its homepage: http://www.speech.kth.se/NICO/index.html

CHAPTER 3

METHODOLOGY AND TOOLS

3.1 Fundamentals to speech recognition

Speech recognition is basically the science of talking with the computer, and having it correctly recognized [17]. To elaborate it we have to understand the following terms [4], [13].

3.1.1 Utterances

When user says some things, then this is an utterance [13] in other words speaking a word or a combination of words that means something to the computer is called an utterance. Utterances are then sent to speech engine to be processed.

3.1.2 Pronunciation

A speech recognition engine uses a process word is its pronunciation, that represents what the speech engine thinks a word should sounds like [4]. Words can have the multiple pronunciations associated with them.

3.1.3 Grammar

Grammar uses particular set of rules in order to define the words and phrases that are going to be recognized by speech engine, more concisely grammar define the domain with which the speech engine works [4]. Grammar can be simple as list of words or flexible enough to support the various degrees of variations.

3.1.4 Accuracy

The performance of the speech recognition system is measurable [4]; the ability of recognizer can be measured by calculating its accuracy. It is useful to identify an utterance.

3.1.5 Vocabularies

Vocabularies are the list of words that can be recognized by the speech recognition engine [4]. Generally the smaller vocabularies are easier to identify by a speech recognition engine, while a large listing of words are difficult task to be identified by engine.

3.1.6 Training

Training can be used by the users who have difficulty of speaking or pronouncing certain words, speech recognition systems with training should be able to adapt.

3.2 Tools

1. Smartdraw2000 (For drawing the Gantt chart and Speech Recognition Model)
2. Visual Paradigm for UML 7.1 (for Use case and Activity Diagram)
3. Ms-Paint
4. Notepad
5. Command Prompt
6. Java development kit 1.6
7. Office 2007 (Documentation)

3.3 Methodology

As an emerging technology, not all developers are familiar with speech recognition technology. While the basic functions of both speech synthesis and speech recognition takes only few minutes to understand (after all, most people learn to speak and listen by age two), there are subtle and powerful capabilities provided by computerized speech that developers will want to understand and utilize.

Despite very substantial investment in speech technology research over the last 40 years, speech synthesis and speech recognition technologies still have significant limitations. Most importantly, speech technology does not always meet the high expectations of users familiar with natural human-to-human speech communication. Understanding the limitations - as well as the strengths - is important for effective use of speech input and output in a user interface and for understanding some of the advanced features of the Java Speech API.

An understanding of the capabilities and limitations of speech technology is also important for developers in making decisions about whether a particular application will benefit from the use of speech input and output.

3.3.1 Speech Synthesis

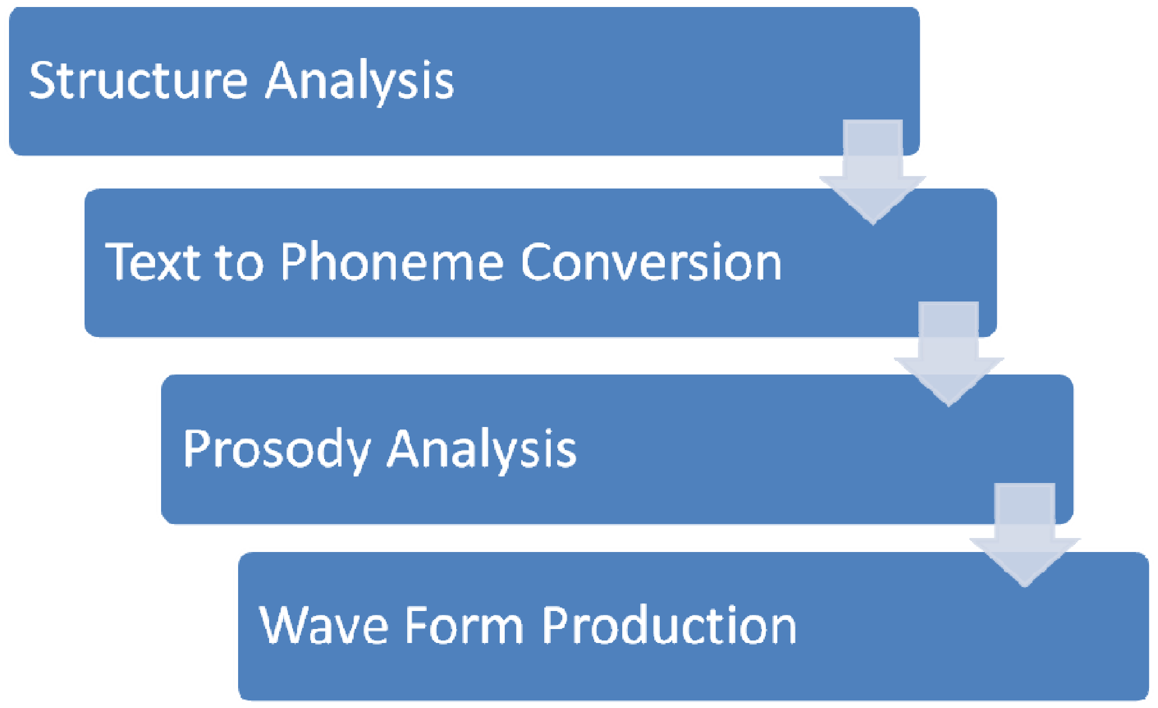


Fig: 3.1 Speech Synthesis

A speech synthesizer converts written text into spoken language. Speech synthesis is also referred to as *text-to-speech* (TTS) conversion.

The major steps in producing speech from text are as follows:

- *Structure analysis*: process the input text to determine where paragraphs, sentences and other structures start and end. For most languages, punctuation and formatting data are used in this stage.

- *Text pre-processing*: analyze the input text for special constructs of the language. In English, special treatment is required for abbreviations, acronyms, dates, times, numbers, currency amounts, email addresses and many other forms. Other languages need special processing for these forms and most languages have other specialized requirements.

The remaining steps convert the spoken text to speech.

- *Text-to-phoneme conversion*: convert each word to *phonemes*. A phoneme is a basic unit of sound in a language. US English has around 45 phonemes including the consonant and vowel sounds. For example, "times" is spoken as four phonemes "t ay m s". Different languages have different sets of sounds (different phonemes). For example, Japanese has fewer phonemes including sounds not found in English, such as "ts" in "tsunami".

- *Prosody analysis*: process the sentence structure, words and phonemes to determine appropriate *prosody* for the sentence. Prosody includes many of the features of speech other than the sounds of the words being spoken. This includes the pitch (or melody), the timing (or rhythm), the pausing, the speaking rate, the emphasis on words and many other features. Correct prosody is important for making speech sound right and for correctly conveying the meaning of a sentence.

- *Waveform production*: finally, the phonemes and prosody information are used to produce the audio waveform for each sentence. There are many ways in which the speech can be produced from the phoneme and prosody information. Most current systems do it in one of two ways: *concatenation* of chunks of recorded human speech, or *formant synthesis* using signal processing techniques based on knowledge of how phonemes sound and how prosody affects those phonemes. The details of waveform generation are not typically important to application developers.

Speech Synthesis: javax.speech.synthesis

A speech synthesizer is a speech engine that converts text to speech. The javax.speech.synthesis package defines the Synthesizer interface to support speech synthesis plus a set of supporting classes and interfaces.

As a type of speech engine, much of the functionality of a Synthesizer is inherited from the Engine interface in the javax.speech package and from other classes and interfaces in that package.

- *Create:* The Central class of javax.speech package is used to obtain a speech synthesizer by calling the createSynthesizer method. The SynthesizerModeDesc argument provides the information needed to locate an appropriate synthesizer. In this example a synthesizer that speaks English is requested.

- *Allocate and Resume:* allocate and resume methods prepare the Synthesizer to produce speech by allocating all required resources and putting it in the RESUMED state.

- *Generate:* The speakPlainText method requests the generation of synthesized speech from a string.

- *Deallocate:* The waitEngineState method blocks the caller until the Synthesizer is in the QUEUE_EMPTY state - until it has finished speaking the text. The deallocate method frees the synthesizer's resources.

3.3.2 Synthesizer as an Engine

The basic functionality provided by a Synthesizer is speaking text, management of a queue of text to be spoken and producing events as these functions proceed. The Synthesizer interface extends the Engine interface to provide this functionality.

The following is a list of the functionality that the javax.speech.synthesis package inherits from the javax.speech package and outlines some of the ways in which that functionality is specialized.

- The properties of a speech engine defined by the EngineModeDesc class apply to synthesizers. The SynthesizerModeDesc class adds information about synthesizer voices

- Synthesizers are searched, selected and created through the Central class in the javax.speech package.

- Synthesizers inherit the basic state system of an engine from the Engine interface. The basic engine states are ALLOCATED, DEALLOCATED, ALLOCATING_RESOURCES and DEALLOCATING_RESOURCES for allocation state, and PAUSED and RESUMED for audio output state. The getEngineState method and other methods are inherited for monitoring engine state. An EngineEvent indicates state changes.
- Synthesizers produce all the standard engine events The javax.speech.synthesis package also extends the EngineListener interface as SynthesizerListener to provide events that are specific to synthesizers.

3.3.3 Selecting Voices

Most speech synthesizers are able to produce a number of voices. In most cases voices attempt to sound natural and human, but some voices may be deliberately mechanical or robotic.

The Voice class is used to encapsulate the four features that describe each voice: voice name, gender, age and speaking style. The voice name and speaking style are both String objects and the contents of those strings are determined by the synthesizer. Typical voice names might be "Victor", "Monica", "Ahmed", "Jose", "My Robot" or something completely different. Speaking styles might include "casual", "business", "robotic" or "happy" (or similar words in other languages) but the API does not impose any restrictions upon the speaking styles. For both voice name and speaking style, synthesizers are encouraged to use strings that are meaningful to users so that they can make sensible judgements when selecting voices.

By contrast the gender and age are both defined by the API so that programmatic selection is possible. The gender of a voice can be GENDER_FEMALE, GENDER_MALE, GENDER_NEUTRAL or GENDER_DONT_CARE. Male and female are hopefully self-explanatory. Gender neutral is intended for voices that are not clearly male or female such as some robotic or artificial voices. The "don't care" values are used when selecting a voice and the feature is not relevant.

The age of a voice can be AGE_CHILD (up to 12 years), AGE_TEENAGER (13-19), AGE_YOUNGER_ADULT (20-40), AGE_MIDDLE_ADULT (40-60), AGE_OLDER_ADULT (60+), AGE_NEUTRAL, and AGE_DONT_CARE.

Both gender and age are OR'able values for both applications and engines. For example, an engine could specify a voice as:

```
Voice("name", GENDER_MALE, AGE_CHILD | AGE_TEENAGER, "style");
```

In the same way that mode descriptors are used by engines to describe themselves and by applications to select from amongst available engines, the Voice class is used both for description and selection. The match method of Voice allows an application to test whether an engine-provided voice has suitable properties.

3.3.4 Speech Recognition

Speech recognition we mean that converting speech into text for this purpose we have use the ocvolume speech recognition engine which is entirely built in java and has provide good results. For using this engine firstly we need to make the object of ocvolume through the constructor.

ocvolume(java.lang.String dict,
java.lang.String folder)

constructor to create a speech recognition engine using VQ for recognition

Parameters:

dict - file path of the dictionary file that contains all the words that the engine can recognize

folder - path of the folder where *.vq are located

after that we have to make the object of mic input class by using the following method

micInput():

Constructor used to create mic object

Afterwards by using the following method we can recognize the spoken words

byteArrayComplete():

This method return true if a word is stored in the buffer

removeOldWord():

This method remove the first element in the buffer

newWord():

This method reads the next element in the word buffer

run():

Starts recording from the microphone

stopRecord():

Stops the recording

setContinuous():

sets   the recording method to continuous

setDiscrete():

Sets the recording method to discrete

3.3 Use case diagram

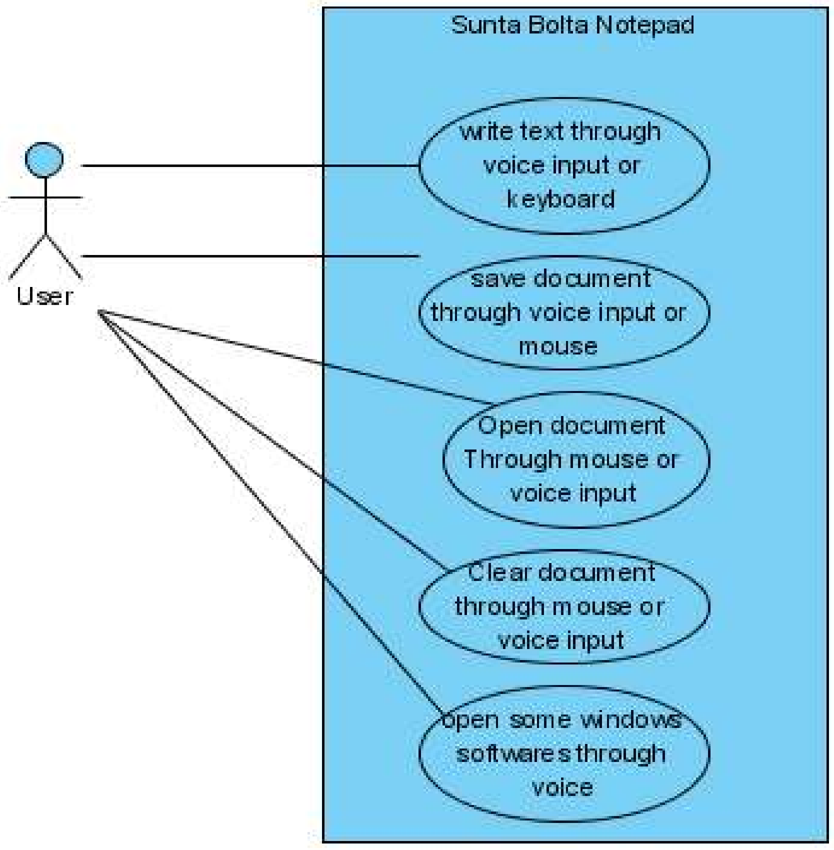

Sunta Bolta Notepad
User
write text through voice input or keyboard
save document through voice input or mouse
Open document Through mouse or voice input
Clear document through mouse or voice input
open some windows softwares through voice

Fig: 3.2 Use case diagram

3.4 Activity Diagram

Saving the document

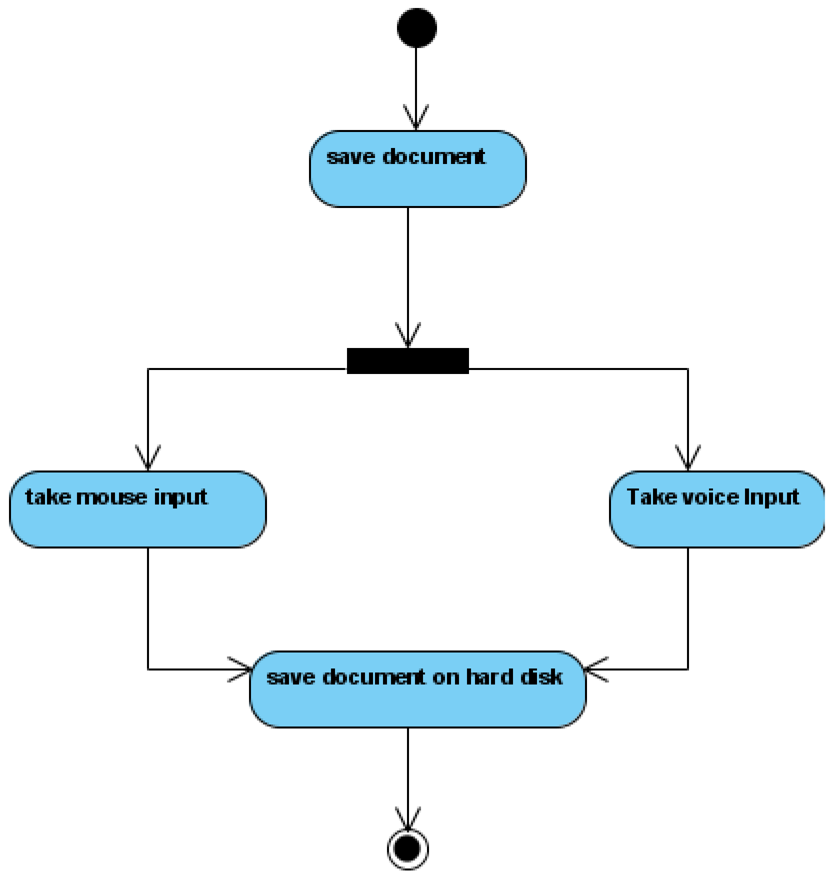


Fig: 3.3 Saving the document

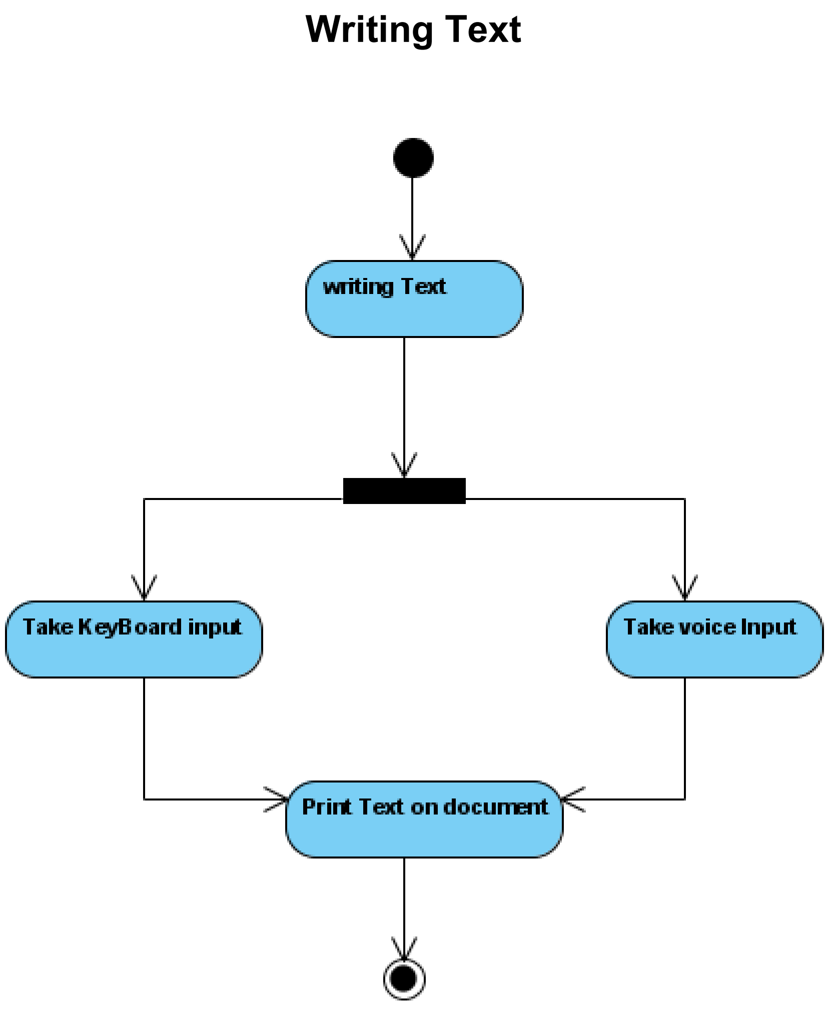


Fig: 3.4 Writing Text

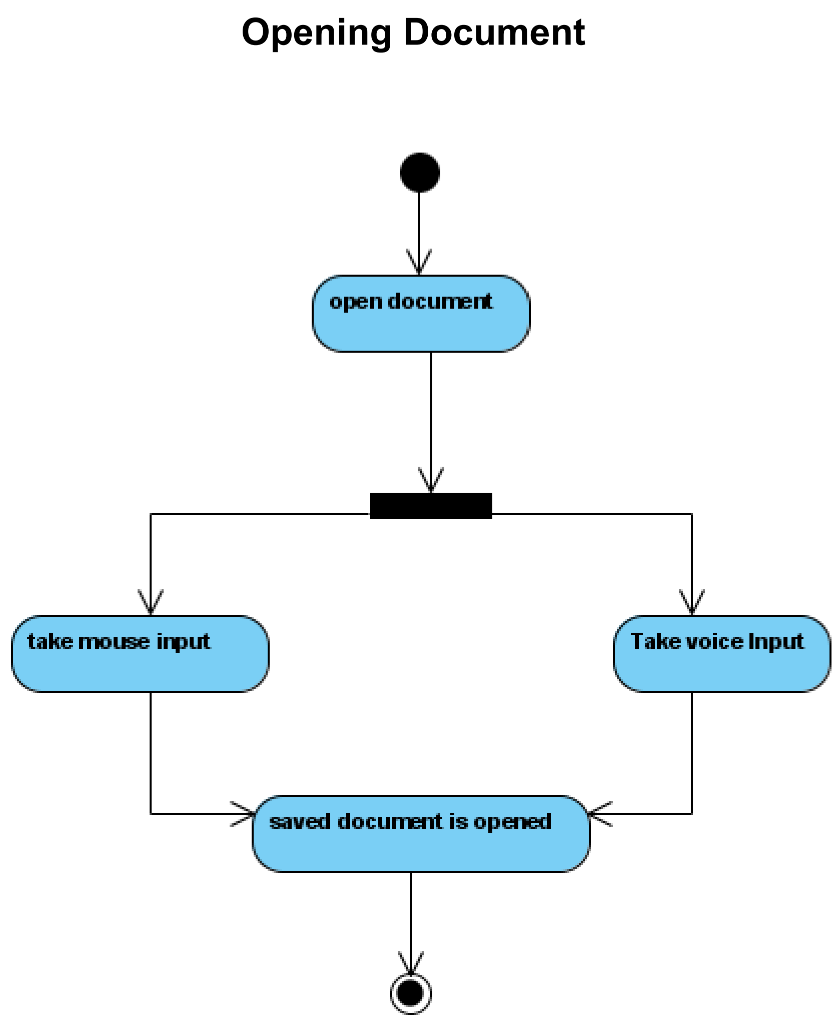


Fig: 3.5 Opening Document

Clearing Document

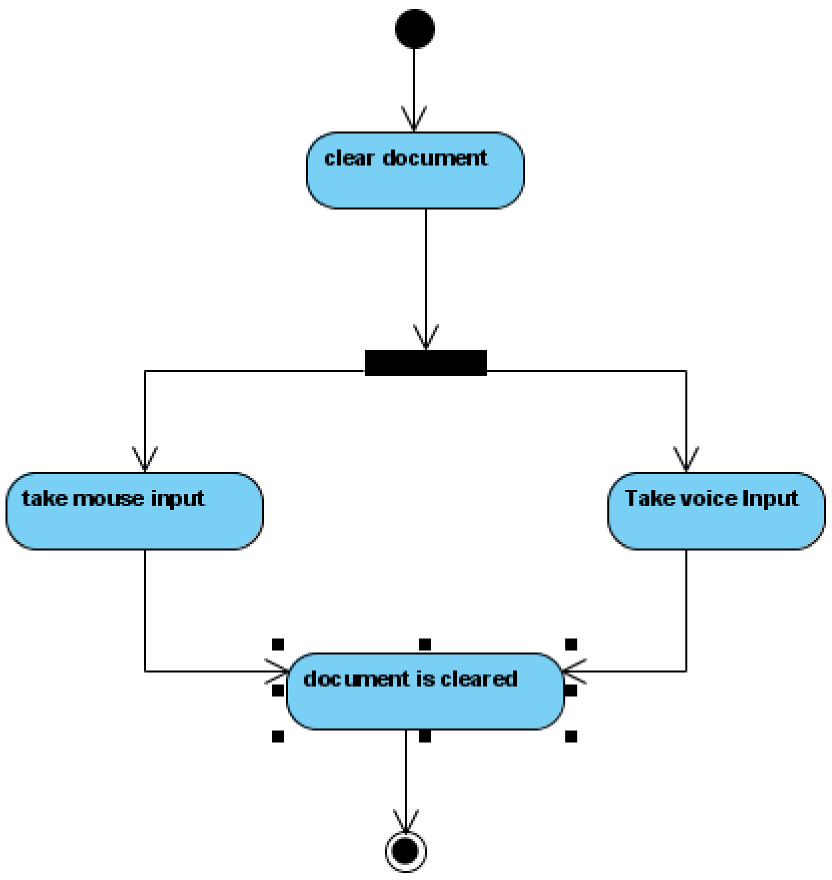


Fig: 3.6 Clearing Document

Opening System Software

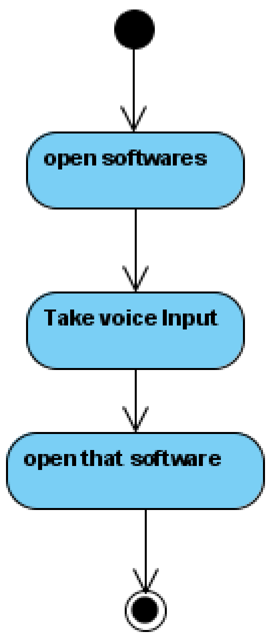


Fig: 3.7 Opening System software

CHAPTER 4

IMPLEMENTATION AND TESTING

4.1 System requirements

4.1.1 Minimum requirements

- Pentium 200 MHz processor
- 64 MB of RAM
- Microphone
- Sound card

4.1.2 Best requirements

- 1.6 GHz Processor
- 128 MB or more of RAM
- Sound cards with very clear signals
- High quality microphones

4.2 Hardware Requirements

Sound cards

Speech requires relatively low bandwidth, high quality 16 bit sound card will be better enough to work [13]. Sound must be enabled, and proper driver should be installed. Sound cards with the 'cleanest' A/D (analog to digital) conversions are recommended, but most often the clarity of the digital sample is more dependent on the microphone quality and even more dependent on the environmental noise. Some speech recognition systems might require specific sound cards.

Microphones

A quality microphone is key when utilizing the speech recognition system. Desktop microphones are not suitable to continue with speech recognition system, because they have tendency to pick up more ambient noise. The best choice, and most common is the headset style. It allows the ambient noise to be minimized, while allowing you to have the microphone at the tip of your tongue all the time. Headsets are available without earphones and with earphones (mono or stereo).

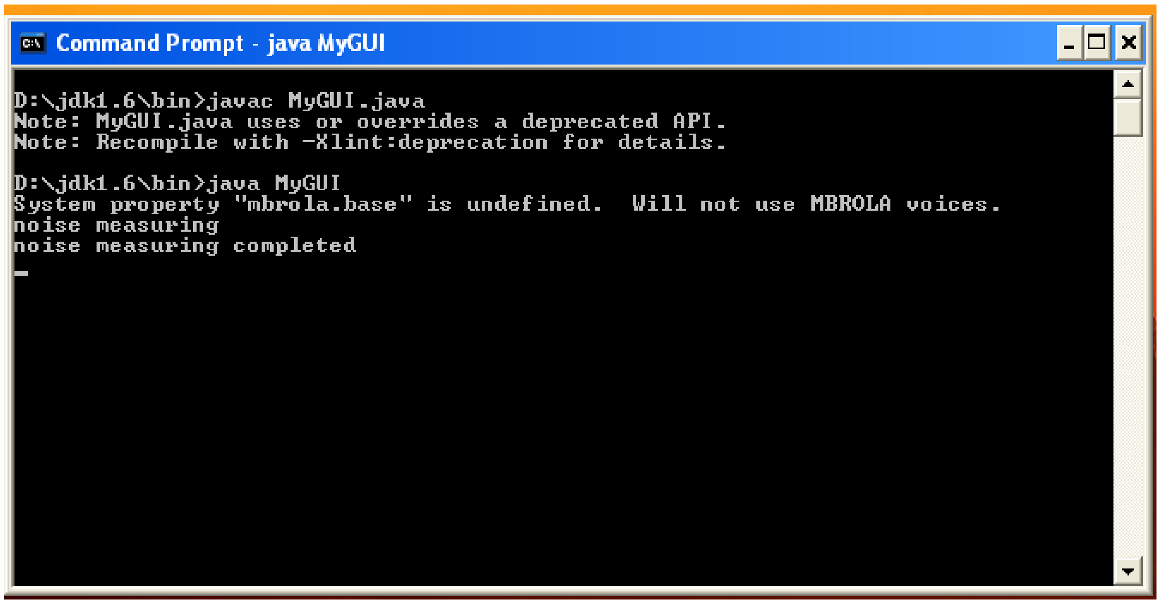
Computer/ Processors

Speech recognition applications can be heavily dependent on processing speed. This is because a large amount of digital filtering and signal processing can take place in ASR.

4.3 Interfaces

Opening Software

Software is opened by using command prompt, after selecting JDK, bin Directory respectively, code file is compiled and then run.



```
D:\jdk1.6\bin>javac MyGUI.java
Note: MyGUI.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\jdk1.6\bin>java MyGUI
System property "mbrola.base" is undefined.  Will not use MBROLA voices.
noise measuring
noise measuring completed
_
```

Fig: 4.1 Opening Software

Software Screen snapshot

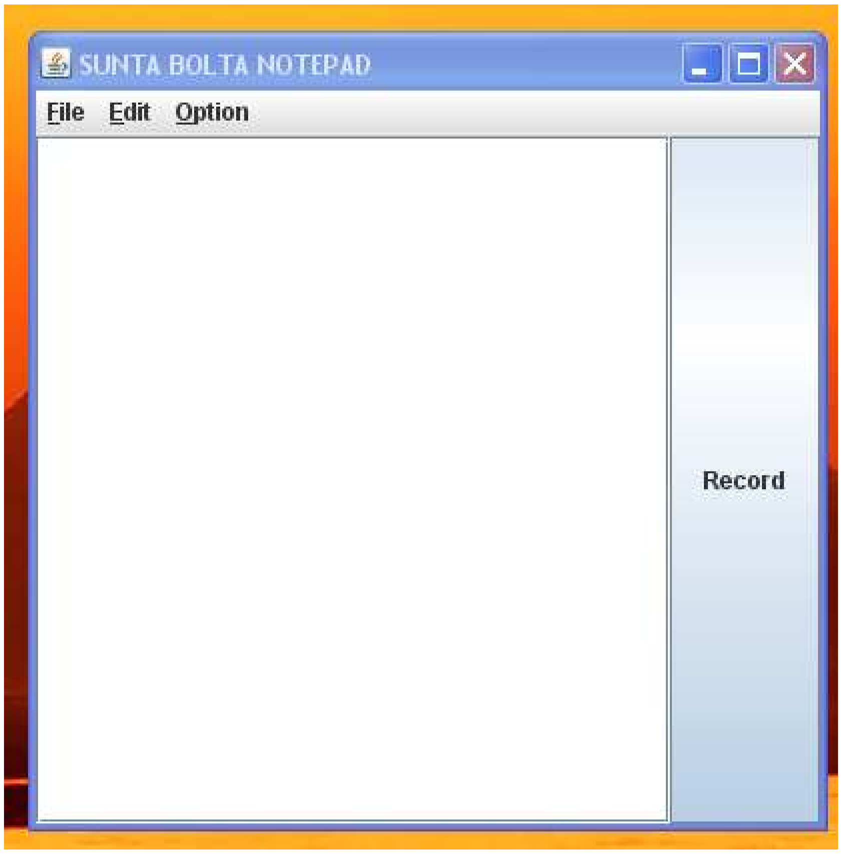


Fig: 4.1 Opening Software

Running File Menu

Software is able to perform different operations, the following screen shot show the contents of file menu, currently 'NEW' menu is active, and when ever new menu is clicked the new text area will be appeared where a use can perform operation on text through both keyboard and voice.

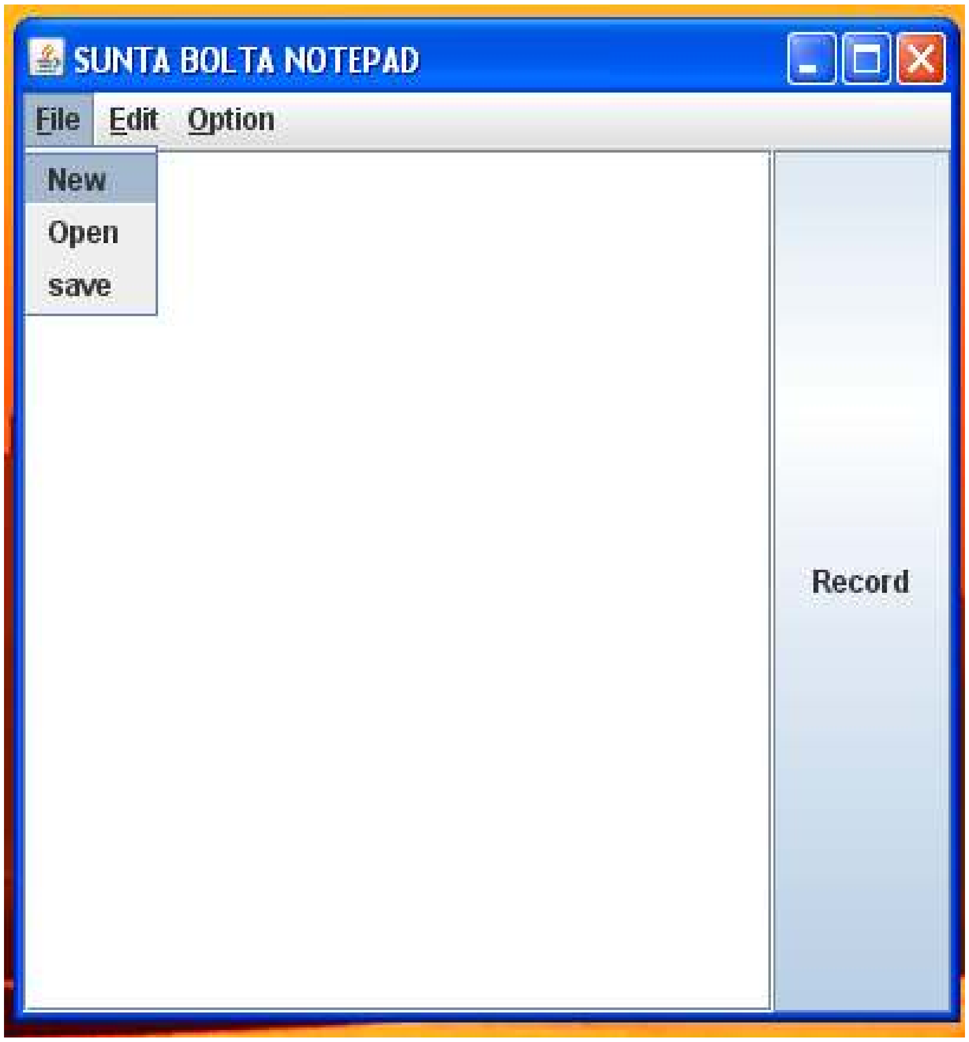
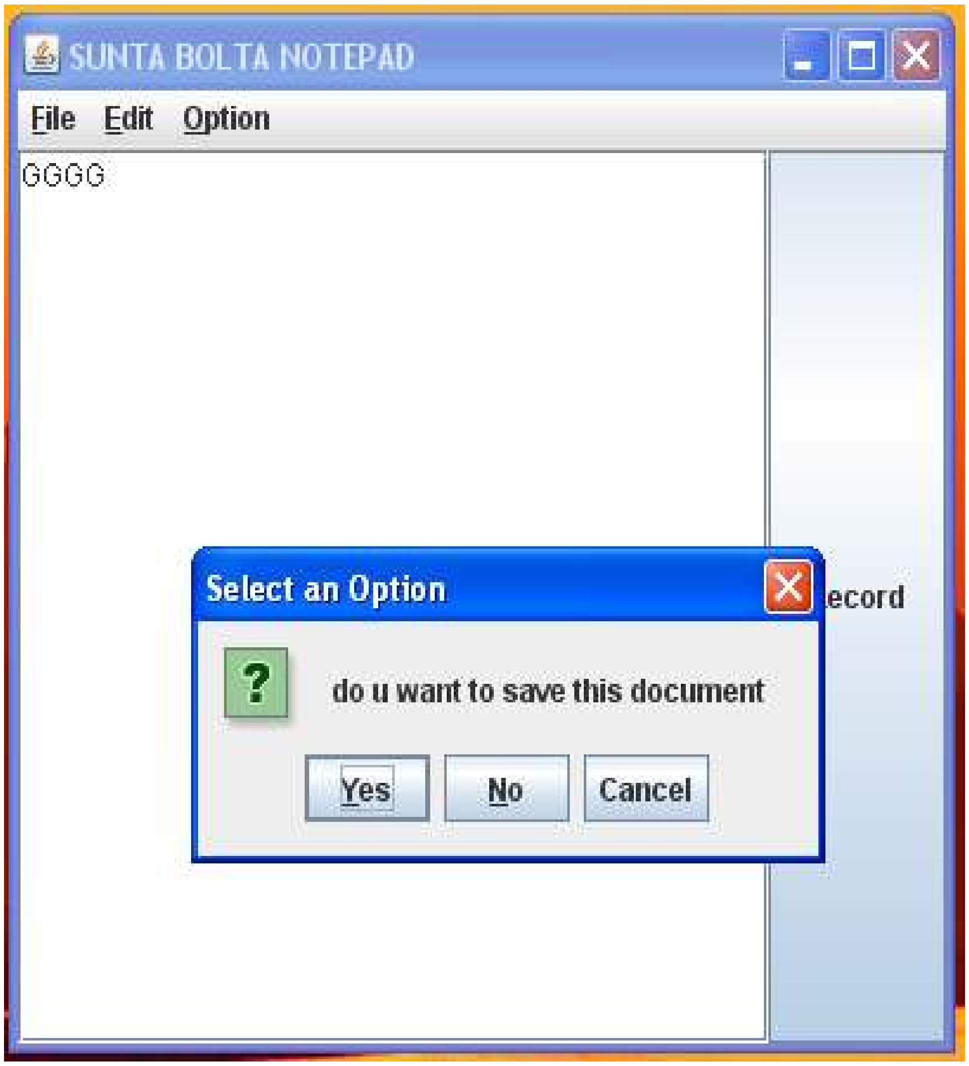


Fig: 4.2 Running file Menu

Saving Document

After writing on the software's text area through keyboard or voice, user can save it in particular directory where he wants to save. The save operation can be performed by both using keyboard/mouse or voice input

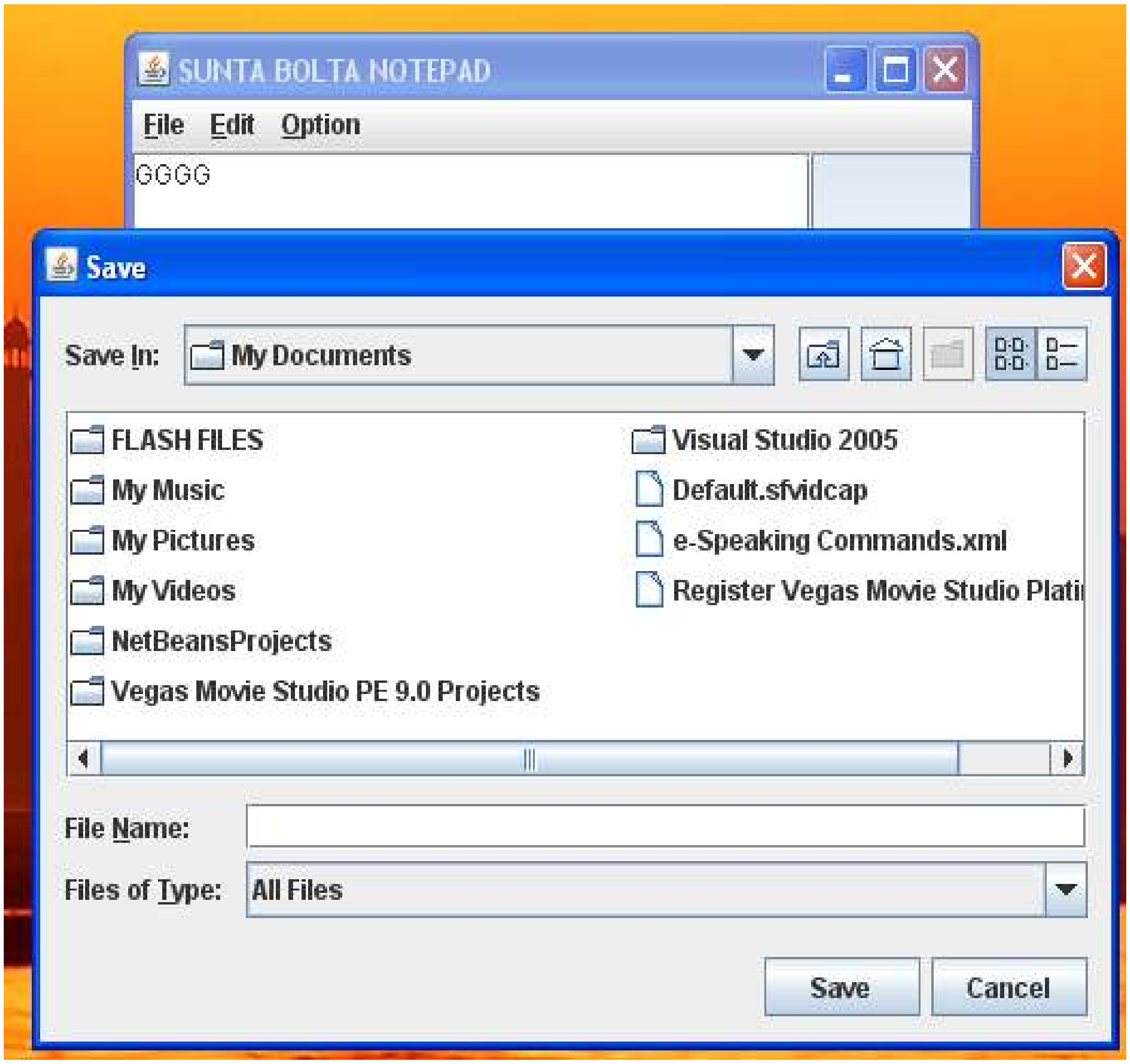


GGGG

do u want to save this document

Fig: 4.3 Saving Document

Particular directory screen shot, here we have assumed that the directory is to be "My Documents"



GGGG

Save

Save In: My Documents

FLASH FILES
My Music
My Pictures
My Videos
NetBeansProjects
Vegas Movie Studio PE 9.0 Projects
Visual Studio 2005
Default.sfvidcap
e-Speaking Commands.xml
Register Vegas Movie Studio Plati

File Name:

Files of Type: All Files

Fig: 4.4 Saving Document

Opening Document

Any document of text format can be opened located any directory in the computer.

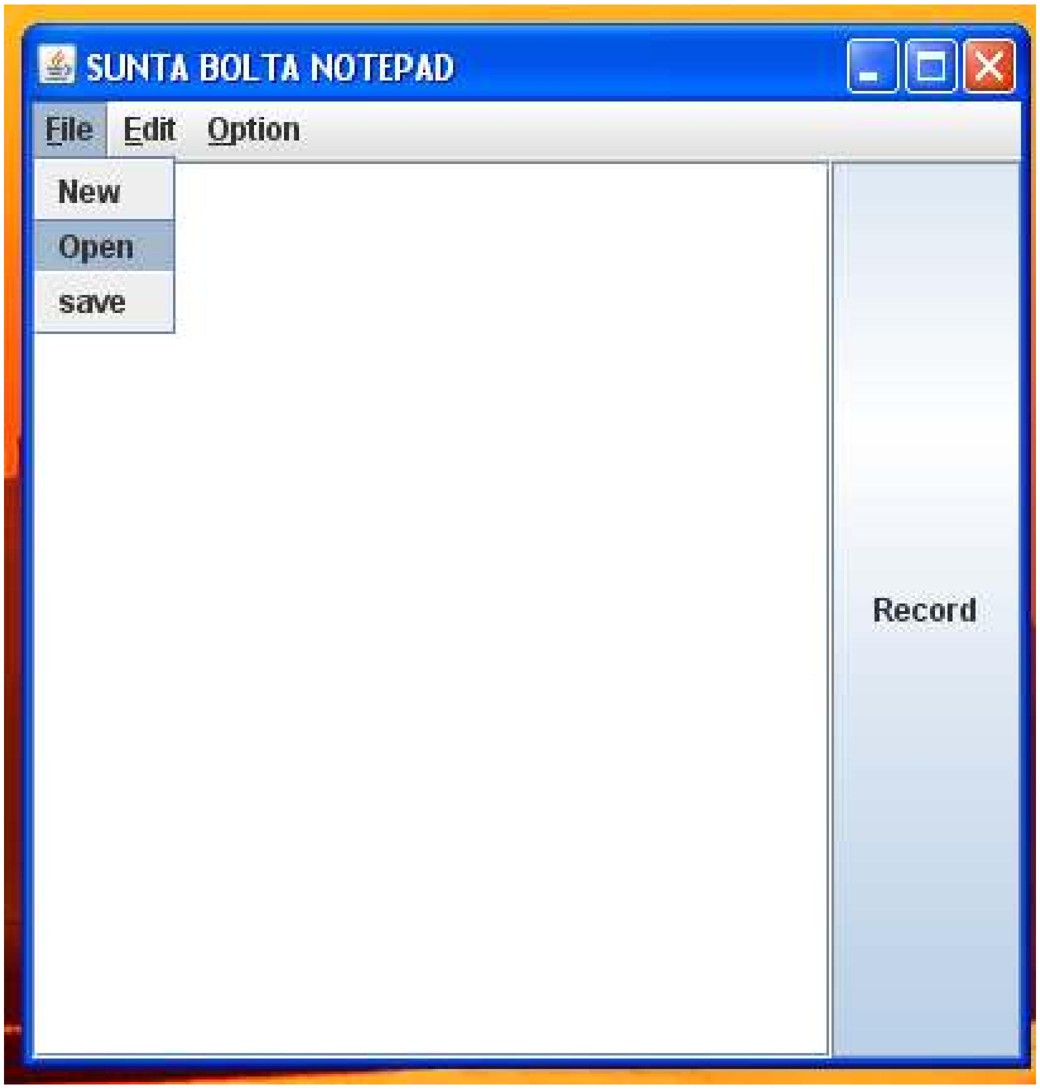
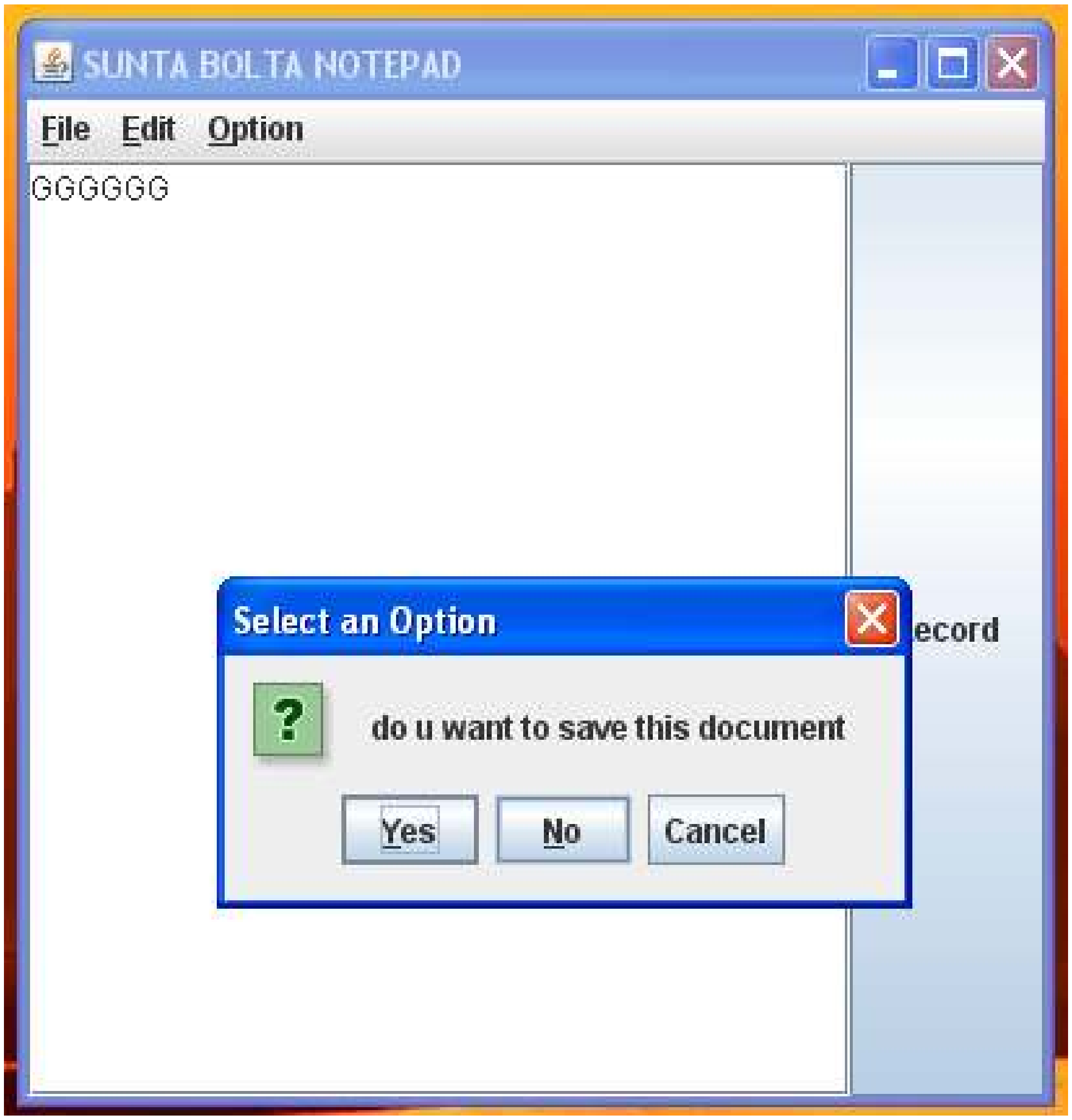


Fig: 4.5 Opening Document

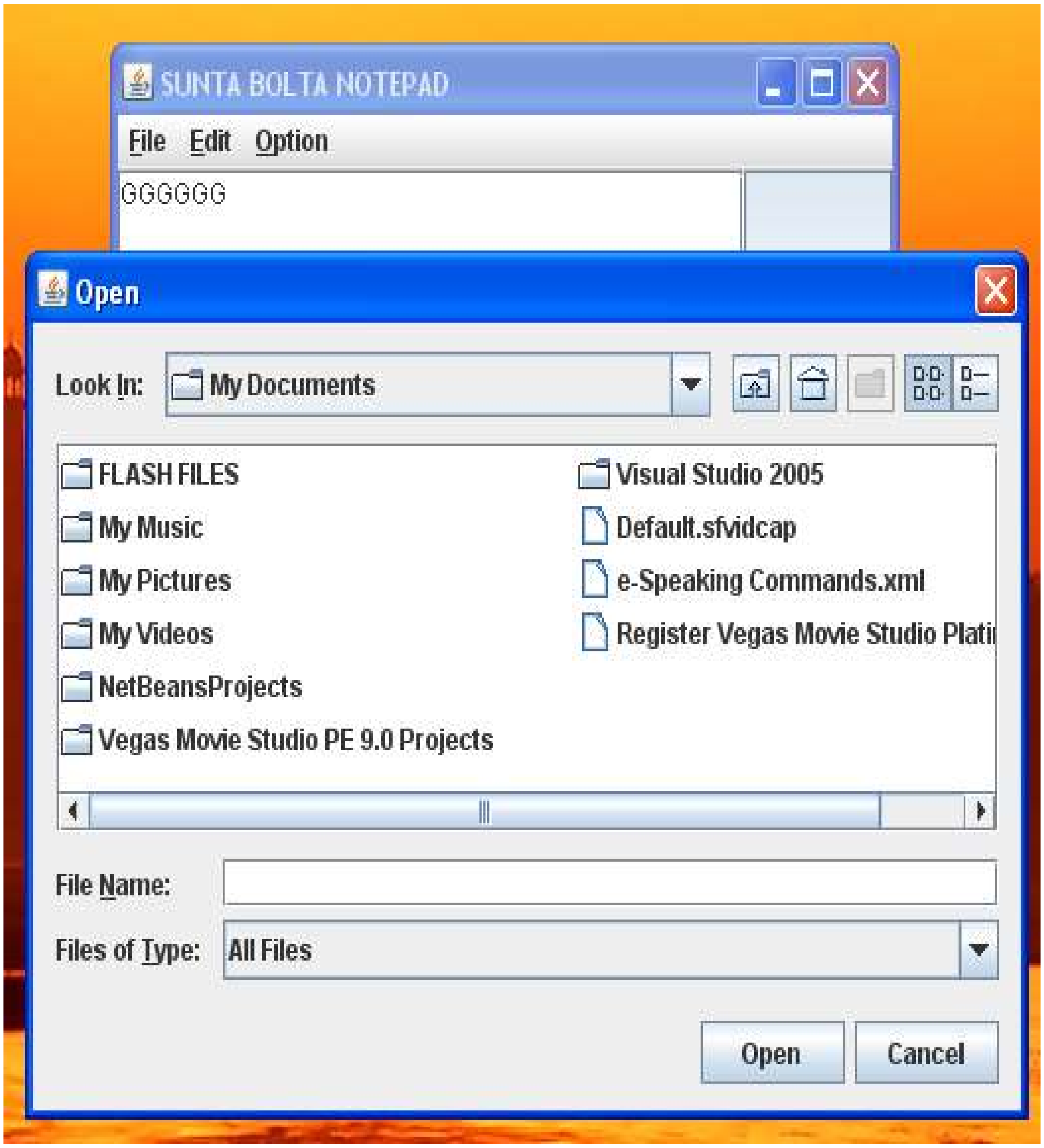
If a user is working on text area of the software and suddenly he want to open a new document then the automatic option will be provided to save current work before opening a new document.



do u want to save this document

Fig: 4.6 Opening Document

After selecting opening option user can browse to it's desired directory. Following is an assumption of my documents directory.



GGGGGG

Look In: My Documents

FLASH FILES
My Music
My Pictures
My Videos
NetBeansProjects
Vegas Movie Studio PE 9.0 Projects
Visual Studio 2005
Default.sfvidcap
e-Speaking Commands.xml
Register Vegas Movie Studio Plati

File Name:

Files of Type: All Files

Fig: 4.7 Opening Document

Edit Menu

The edit menu consist a number of options for different operations, cut, copy, paste, select all, font size and font style can be viewed from following screen shot.

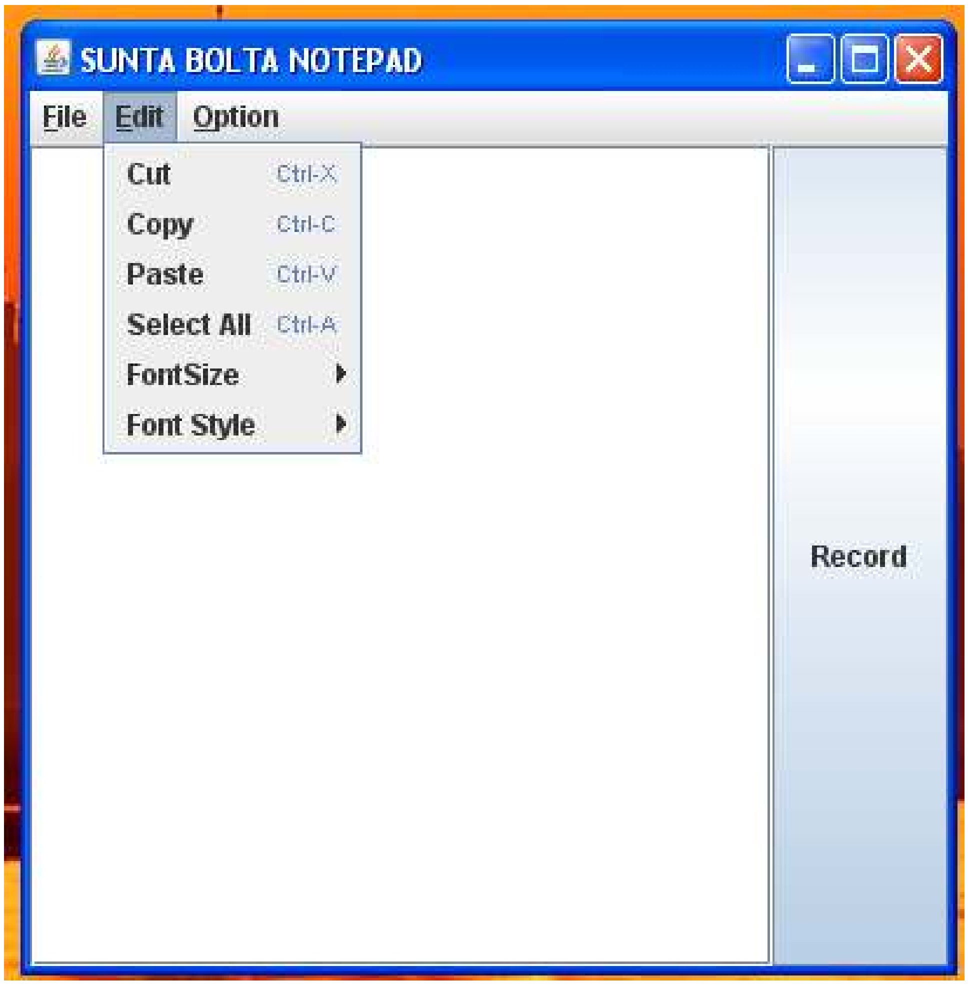


Fig: 4.8 Edit Menu

Cut Option

Cut option in the software is similar to that present in all other. Particular selected text can be deleted from a location and can be pasted any where else in the document.

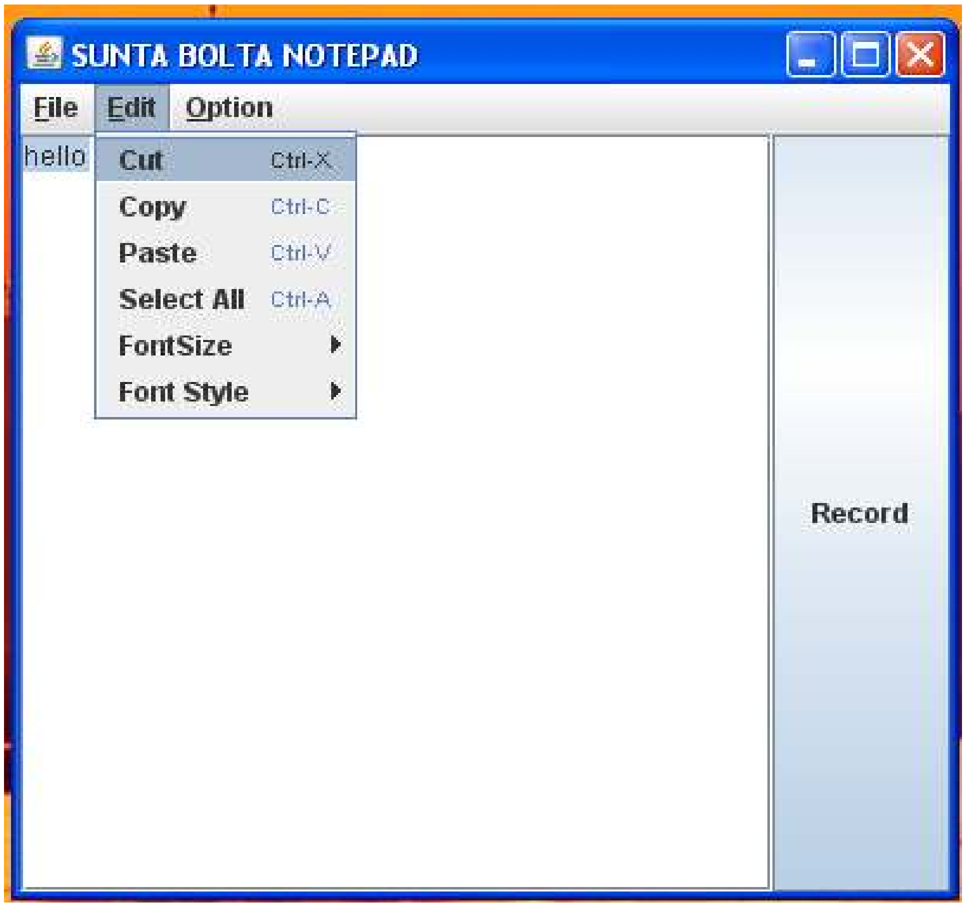
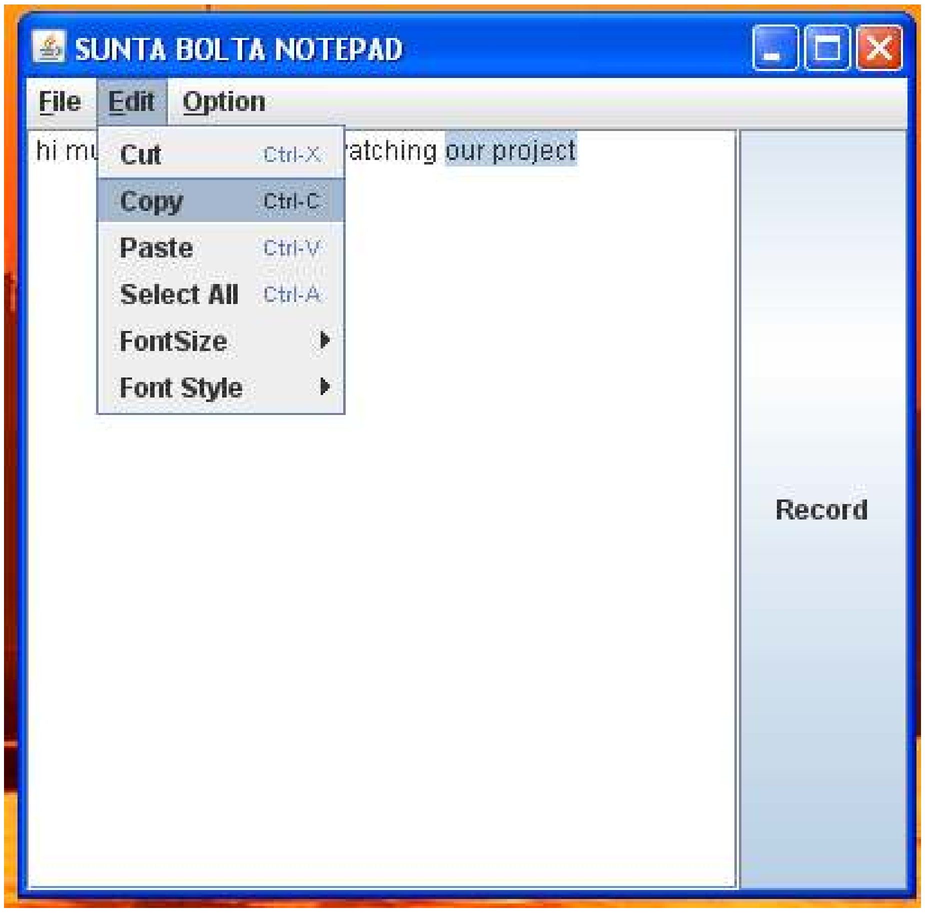


Fig:4.9 Cut Option

Copy Option

Performs the copy option, that is copy text from one location of document to another location

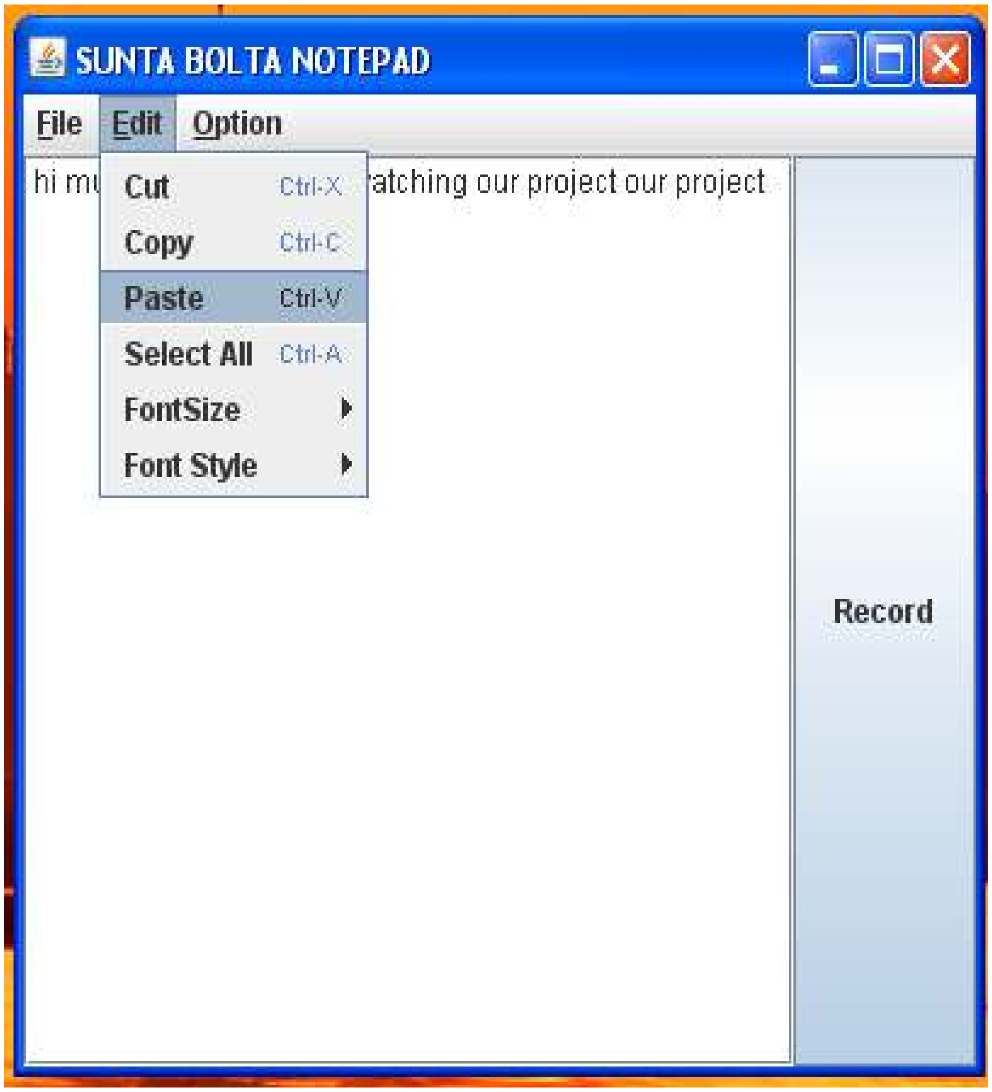


hi mu atching our project

Fig:4.10 Copy Option

Paste

Paste the text onto the selected location of the software in the document, that were copied by using both cut and copy options

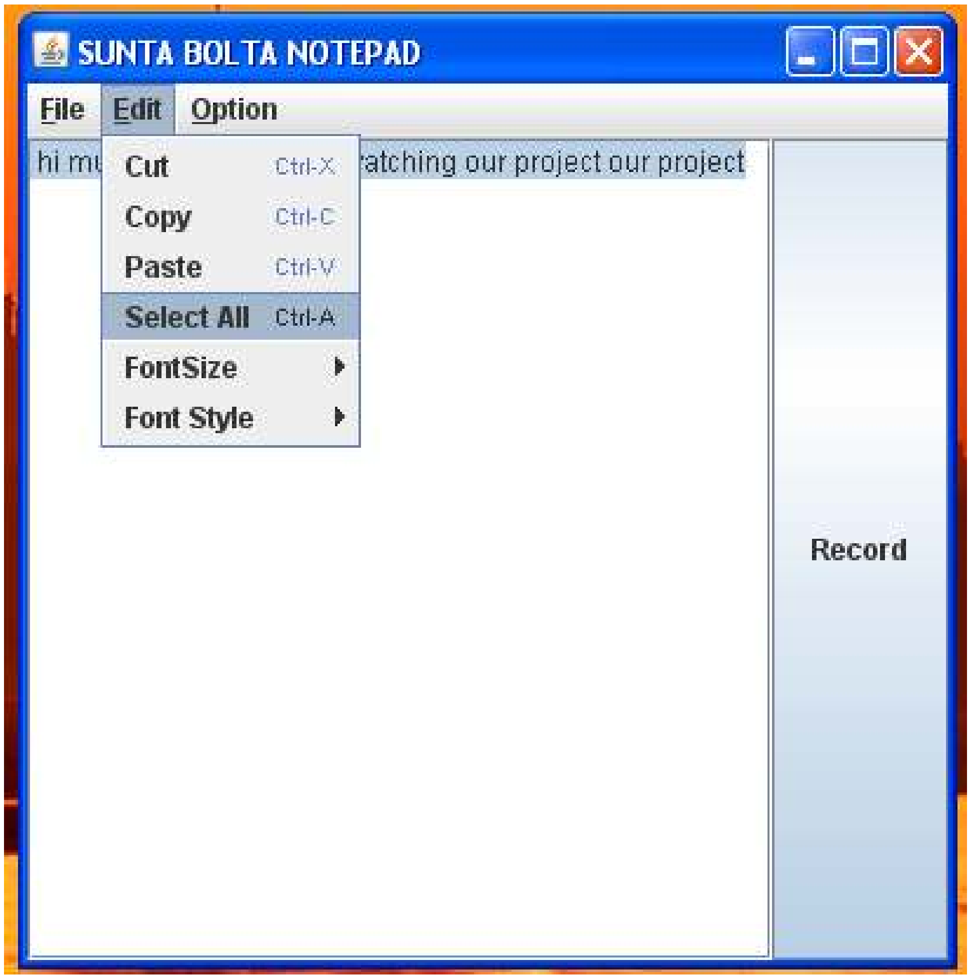


hi m atching our project our project

Fig:4.11 Paste

Select All

Select all the text currently present on the text area of the notepad.

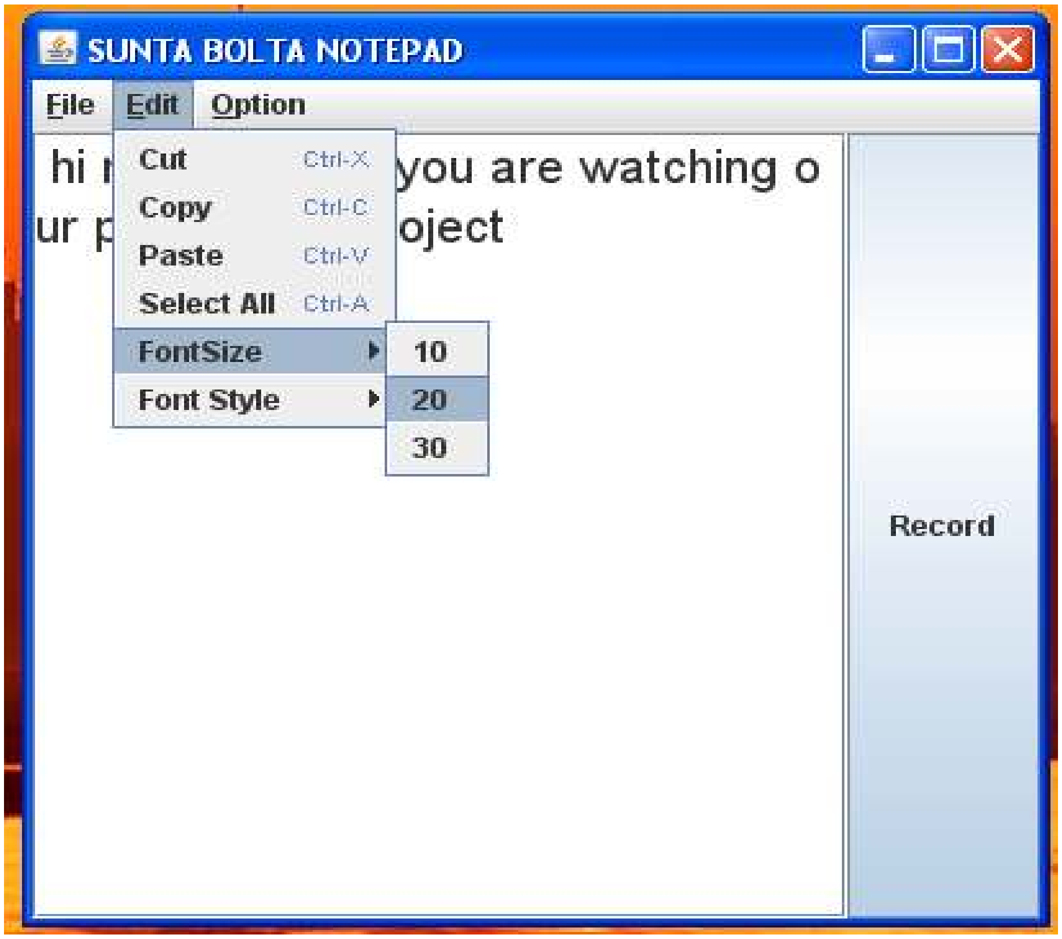


hi mu atching our project our project

Fig:4.12 Select All

Font Size

Sets the selected/modified size of the text

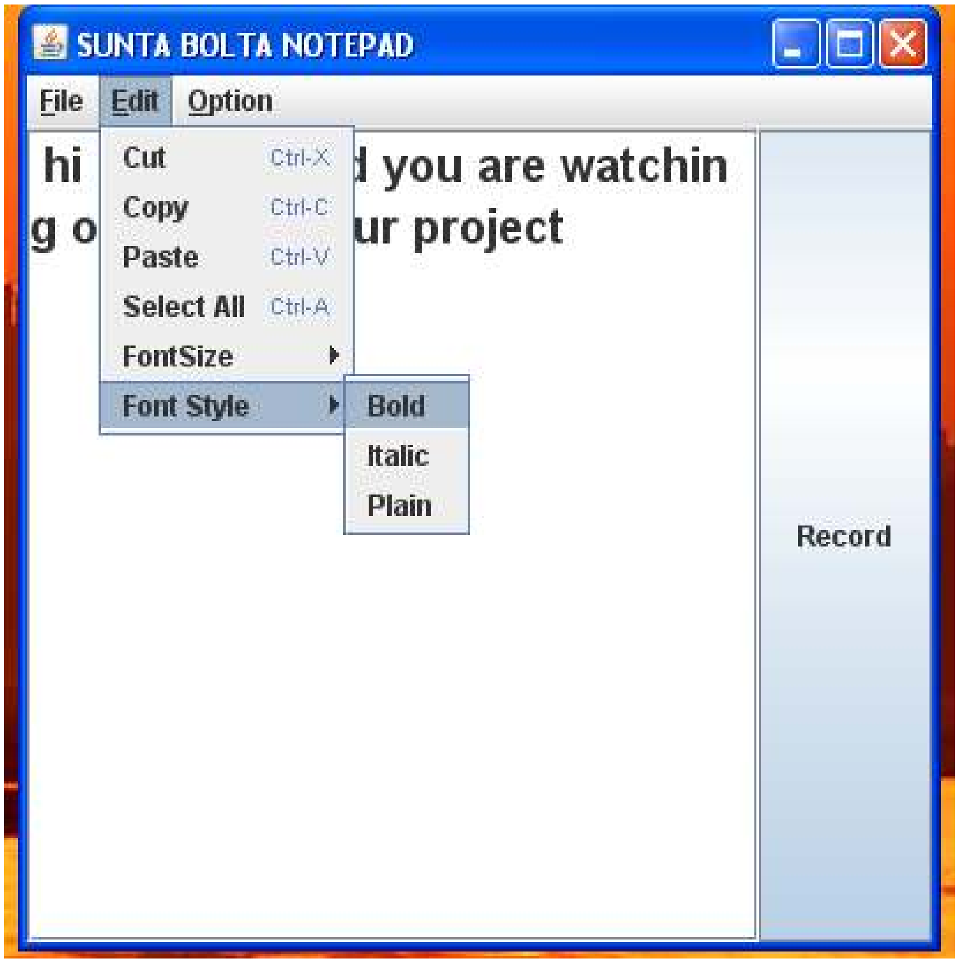


hi r you are watching o
ur p oject

Fig:4.13 Font Size

Font Style

Sets the user specified font style of text



hi ... d you are watchin
g o... ur project

Fig: 4.14 Font Style

Running System Commands

Through this option the software will run different softwares based on the identification of spoken speech.

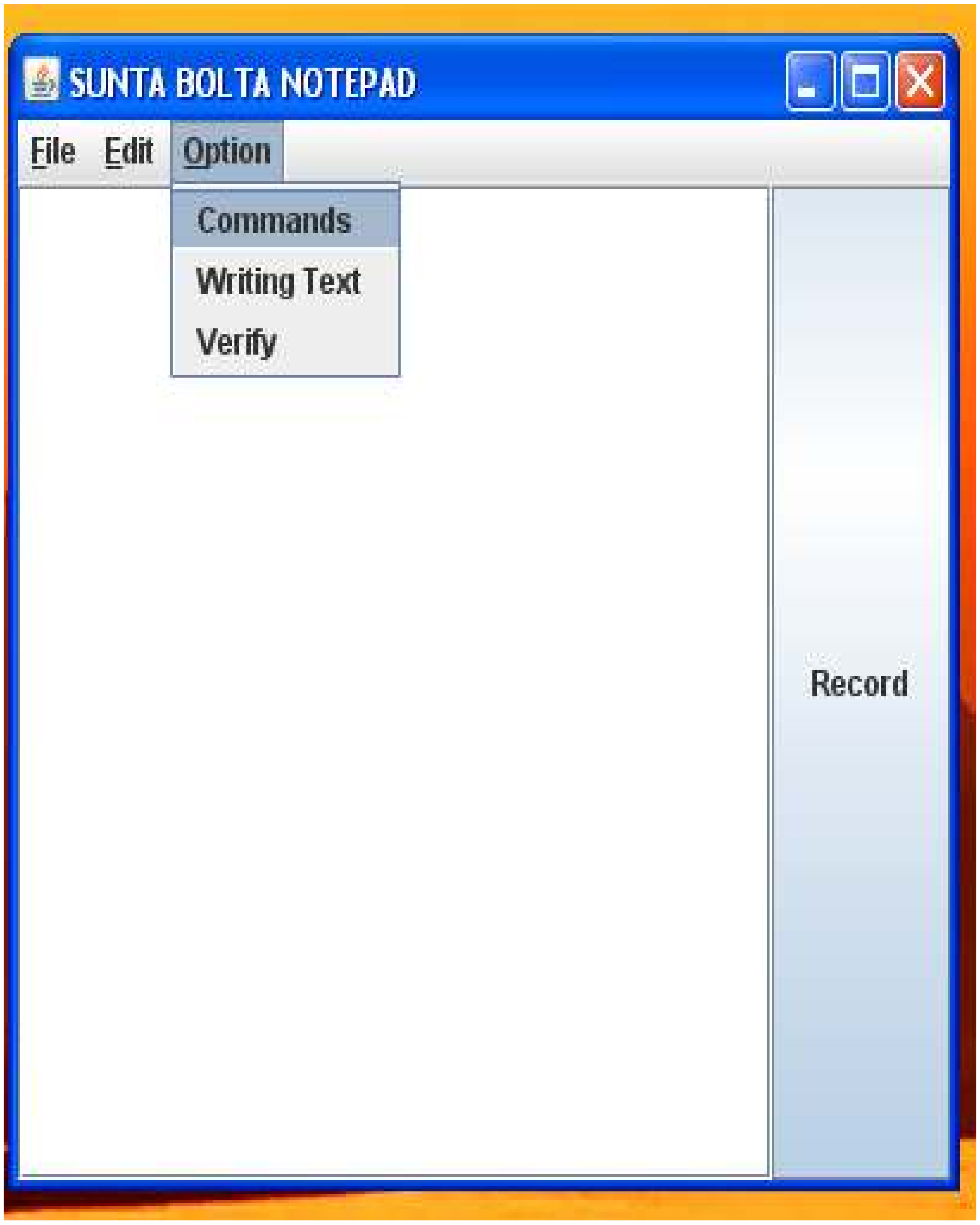
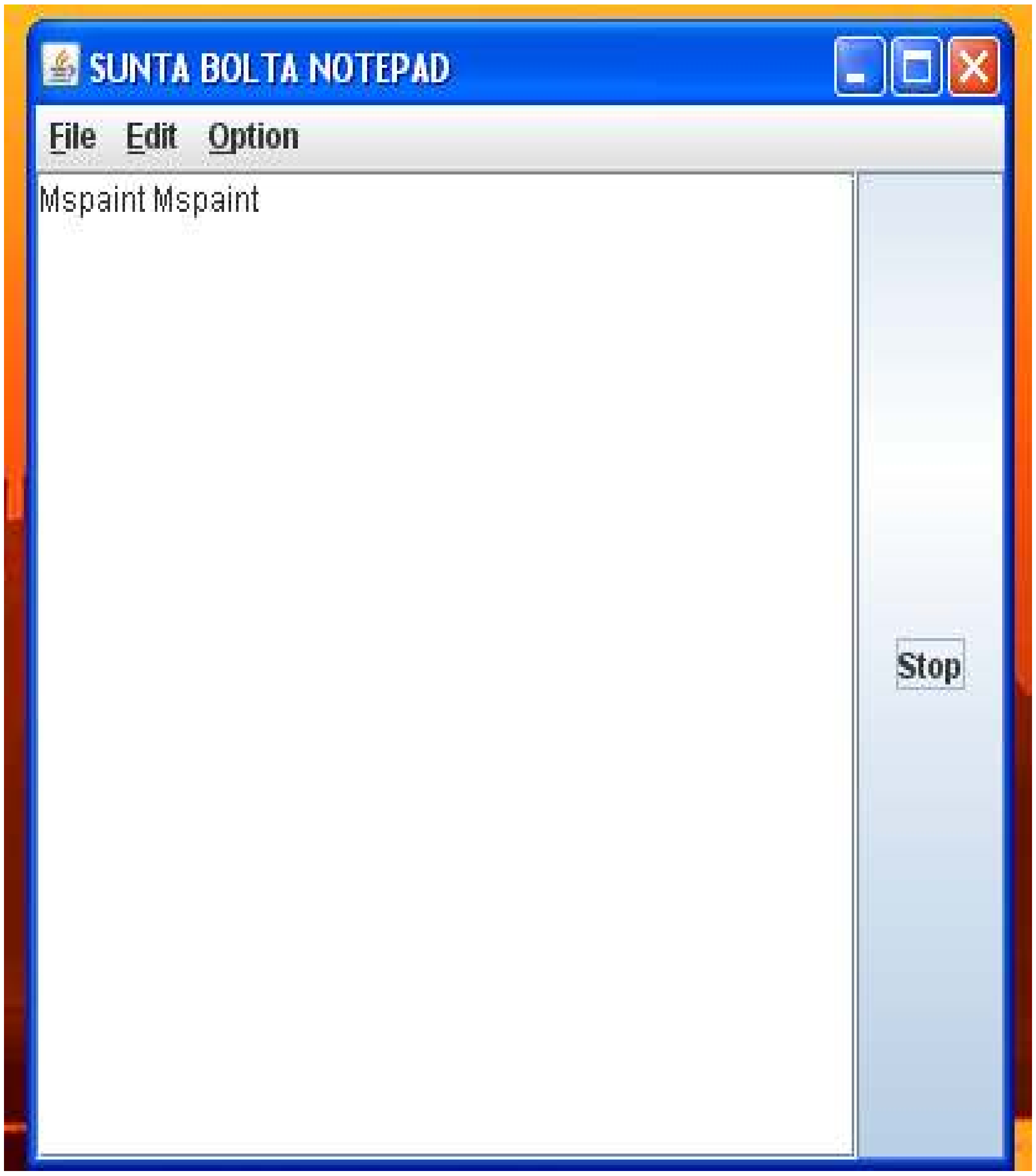


Fig: 4.15 Running System Commands

Text Through Voice

After enabling this option the software would be capable to record human speech and convert it into the text and output it in written form based on identification of input speech.



Mspaint Mspaint

Fig: 4.16 Text Through Voice

Verifying Text

Through this option a user can verify the text; the system will verify text in the form of speech.

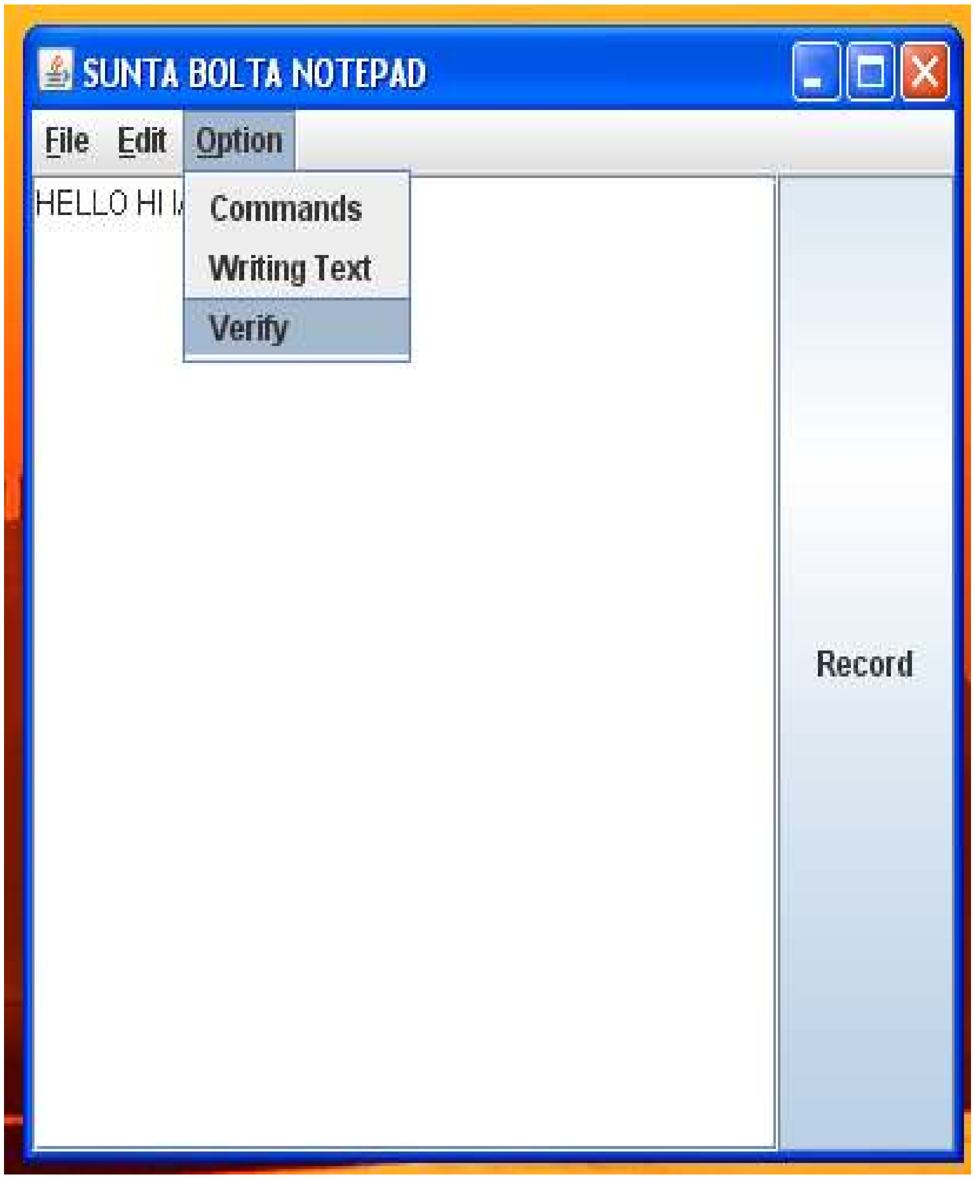


Fig: 4.17 Verifying Text

4.4 Working

This software is designed to recognize the speech and also has the capabilities for speaking and synthesizing means it can convert speech to text and text to speech. This software named ‘SUNTA BOLTA NOTEPAD’ has the capability to write spoken words into text area of notepad, and also can recognize your commands as “save, open, clear” this software is capable of opening windows software such as notepad, ms paint, calculator through voice input.

The synthesize part of this software helps in verifying the various operations done by user such as read out the written text for user also informing that what type of actions a user is doing such as saving a document, opening a new file or opening a file previously saved on hard disk

4.5 Initial Test, Results and discussions

Test	Input signal	Results	Discussions
1.Writing Text	(I) Keyboard	Printed output of the input signal on screen	We Provided input through keyboard and by pressing each printable key and observed that it worked fine as expected
	(II)Voice	Displayed the output of the input signal on screen	While providing voice input to the software it recognized the spoken words in few attempts, this is due to noisy environment, variation in the voice and multiple user factor
2. Running notepad commands (open, save, clear)	(I) Mouse/Keyboard	Properly functioning of open, save and clear to file were observed	we properly run the notepad commands through mouse input and they worked properly fine
	(II) Voice	Properly functioning of open, save and clear to file were observed	we provide voice input to run the notepad commands and they work fine according to the expectations
3. Running System commands (Calculator, Ms-Paint, Notepad)	(I)Voice	System soft wares were opened and Commands worked finely.	By providing voice input to the software for running system commands it worked fine and result in expectations, without repeating the commands twice or thrice.

Table#4.1 Initial Tests, Results and discussions

CHAPTER 5

CONCLUSION

5.1 Advantages of software

- ✓ Able to write the text through both keyboard and voice input.
- ✓ Voice recognition of different notepad commands such as open save and clear.
- ✓ Open different windows soft wares, based on voice input.
- ✓ Requires less consumption of time in writing text.
- ✓ Provide significant help for the people with disabilities.
- ✓ Lower operational costs.

5.2 Disadvantages

- ✓ Low accuracy
- ✓ Not good in the noisy environment

5.3 Future Enhancements

This work can be taken into more detail and more work can be done on the project in order to bring modifications and additional features. The current software doesn't support a large vocabulary, the work will be done in order to accumulate more number of samples and increase the efficiency of the software. The current version of the software supports only few areas of the notepad but more areas can be covered and effort will be made in this regard.

5.4 Conclusion

This Thesis/Project work of speech recognition started with a brief introduction of the technology and its applications in different sectors. The project part of the Report was based on software development for speech recognition. At the later stage we discussed different tools for bringing that idea into practical work. After the development of the software finally it was tested and results were discussed, few deficiencies factors were brought in front. After the testing work, advantages of the software were described and suggestions for further enhancement and improvement were discussed.

REFERENCES

BOOKS

[1] “Speech recognition- The next revolution” 5th edition.

[2] Ksenia Shalonova, “Automatic Speech Recognition” 07 DEC 2007

Source:http://www.cs.bris.ac.uk/Teaching/Resources/COMS12303/lectures/Ksenia_Shalonova-Speech_Recognition.pdf

[4] "Fundamentals of Speech Recognition". L. Rabiner & B. Juang. 1993. ISBN: 0130151572.

[5] "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition". D. Jurafsky, J. Martin. 2000. ISBN: 0130950696.

[15] *B.H. Juang & Lawrence R. Rabiner*, “Automatic Speech Recognition – A Brief History of the Technology Development” 10/08/2004

Source:http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf

[13] Stephen Cook “”Speech Recognition HOWTO” Revision v2.0 April 19, 2002

Source: http://www.scribd.com/doc/2586608/speechrecognitionhowto

INTERNET

[3] http://www.abilityhub.com/speech/speech-description.htm

[7] Charu Joshi "Speech Recognition"
Source: http://www.scribd.com/doc/2586608/speechrecognition.pdf
Date Added 04/21/2008

[8] John Kirriemuir "Speech recognition technologies" March 30th2003

[9]http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speechrecognition.htm/printable last updated: 30th October 2009

[10] http://www.jisc.ac.uk/media/documents/techwatch/ruchi.pdf

[11] http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition3.ht

[12] http://en.wikipedia.org/wiki/Speech_recognition Visited: 12NOV2009

APPENDICES

Appendix B

Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import org.oc.ocvolume.*;
import org.oc.ocvolume.audio.*;

import javax.speech.Central;
import javax.speech.synthesis.Synthesizer;
import javax.speech.synthesis.SynthesizerModeDesc;
import javax.speech.synthesis.Voice;
import java.util.*;


public class MyGUI implements ActionListener,Runnable{


        Runtime r=Runtime.getRuntime();

        JFrame myFrame;

        JTextArea myArea;

        Font font;

        JTextField myText;

//menu section

        JMenuBar menu;

        JMenu file;

        JMenu edit;

        JMenu fontStyle;

        JMenuItem new1;

        JMenuItem open;

        JMenuItem save;
```

```
          JButton record;

          JMenu option;

          JMenuItem commands;

          JMenuItem wText;

          JMenuItem verify;

          JMenuItem bold;

          JMenuItem italic;

          JMenuItem plain;

          JMenu fontSize;

          JMenuItem f10;

          JMenuItem f20;

          JMenuItem f30;

          JMenuItem copy;

          JMenuItem paste;

          JMenuItem cut;

          JMenuItem sel;


//synthesizer part

          String voiceName;

          SynthesizerModeDesc desc;

          Synthesizer synthesizer;

          Voice[] voices;

          Voice voice;
```

```
        boolean comm=false;

        JFileChooser fc=new JFileChooser();

        private ocvolume engine = new ocvolume("dict", "");

        private micInput mic = new micInput();

        private Thread checking;

public void myDesign(){                  //design function

        myFrame=new JFrame("SUNTA BOLTA NOTEPAD");

        Container c=myFrame.getContentPane();

        c.setLayout(new BorderLayout());

        record=new JButton("Record");


        myArea=new JTextArea();

        myArea.setLineWrap(true);

        JScrollPane bar=new JScrollPane(myArea);

        myText=new JTextField();


        font=new Font("abc",10,3);

        menu=new JMenuBar();

        file=new JMenu("File");

        file.setMnemonic('f');


        option=new JMenu("Option");

        option.setMnemonic('o');

        edit=new JMenu("Edit");

        edit.setMnemonic('e');
```

```
new1=new JMenuItem("New");
open=new JMenuItem("Open");
save=new JMenuItem("save");
commands =new JMenuItem("Commands");
wText=new JMenuItem("Writing Text");
verify=new JMenuItem("Verify");

fontStyle=new JMenu("Font Style");
bold=new JMenuItem("Bold");
italic=new JMenuItem("Italic");
plain=new JMenuItem("Plain");

fontSize=new JMenu("FontSize");
f10=new JMenuItem("10");
f20=new JMenuItem("20");
f30=new JMenuItem("30");
copy=new JMenuItem("Copy");
paste=new JMenuItem("Paste");
cut=new JMenuItem("Cut");
sel =new JMenuItem ("Select All");


new1.addActionListener(this);
open.addActionListener(this);
save.addActionListener(this);
record.addActionListener(this);
```

```
        commands.addActionListener(this);

        wText.addActionListener(this);

        verify.addActionListener(this);

        bold.addActionListener(this);

        italic.addActionListener(this);

        plain.addActionListener(this);

        copy.addActionListener(this);

        paste.addActionListener(this);

        cut.addActionListener(this);

        sel.addActionListener(this);


        f10.addActionListener(this);

        f20.addActionListener(this);

        f30.addActionListener(this);


new1.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N,
ActionEvent.CTRL_MASK));

open.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,
ActionEvent.CTRL_MASK));

save.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
ActionEvent.CTRL_MASK));

cut.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X,
ActionEvent.CTRL_MASK));

copy.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C,
ActionEvent.CTRL_MASK));

paste.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V,
ActionEvent.CTRL_MASK));
```

```
sel.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_A,
ActionEvent.CTRL_MASK));



file.add(new1);

file.add(open);

file.add(save);

option.add(commands);

option.add(wText);

option.add(verify);

fontSize.add(f10);

fontSize.add(f20);

fontSize.add(f30);

edit.add(cut);

edit.add(copy);

edit.add(paste);

edit.add(sel);

fontStyle.add(bold);

fontStyle.add(italic);

fontStyle.add(plain);


edit.add(fontSize);

edit.add(fontStyle);


menu.add(file);

menu.add(edit);
menu.add(option);

c.add(menu,BorderLayout.NORTH);
c.add(bar,BorderLayout.CENTER);
c.add(record,BorderLayout.EAST);
myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
myFrame.setSize(400,400);
myFrame.setVisible(true);



}//end of Design





MyGUI(){
synthinit();
myDesign();


}//end of the constructor

public void actionPerformed(ActionEvent e){

        if(e.getSource()==sel){

                myArea.selectAll();
        }



    if(e.getSource()==copy){

        myArea.copy();

        }
    if(e.getSource()==paste){

        myArea.paste();

        }
    if(e.getSource()==cut){

        myArea.cut();

        }



    if(e.getSource()==bold){
        int c1=Font.BOLD;
        font=font.deriveFont(c1);
        myArea.setFont(font);

        }
    if(e.getSource()==italic){
        int c1=Font.ITALIC;
        font=font.deriveFont(c1);

        myArea.setFont(font);


        }

if(e.getSource()==plain){

        myArea.setFont(null);


        }


if(e.getSource()==f10){

        float c1=(float)10.0;


        font=font.deriveFont(c1);

        myArea.setFont(font);


        }

if(e.getSource()==f20){

        float c1=(float)20.0;


        font=font.deriveFont(c1);

        myArea.setFont(font);


        }

if(e.getSource()==f30){

        float c1=(float)30.0;


        font=font.deriveFont(c1);
```

```
            myArea.setFont(font);

            }


    if(e.getSource()==verify){

            String str = new String(myArea.getText());//String to be spoken

            synthesizer.speakPlainText(str, null);

            }


    if(e.getSource()==commands){

            comm=true;


            }//end of commands event

    if(e.getSource()==wText){

            comm=false;

            }//end of writing text event




    if(e.getSource()==record){

            if ( record.getLabel().equals("Record") == true ){

        record.setLabel("Stop");

        mic = new micInput();

        mic.start();

       try{
            checking.stop();
```

```
                    }
                    catch(NullPointerException e1){}
                    checking = new Thread(this);
                    checking.start();
                     }

                else if ( record.getLabel().equals("Stop") == true ){
                    record.setLabel("Record");
                    mic.stopRecord();
                }





            }// end of record button event




            if (e.getSource()==new1){



                    if(!myArea.getText().equals("")){


                            String str = new String("do u want to save this document
");//String to be spoken
                            synthesizer.speakPlainText(str, null);

                            int ch=JOptionPane.showConfirmDialog(null,"do u want to save
this document");






                    if(ch==0){






                    saveFile();}}

                            String str = new String("your document is cleared ");//String to be
spoken
```

```
                    synthesizer.speakPlainText(str, null);

                    myArea.setText("");

                    }// end of new1 event


            else if (e.getSource()==open){

                    if(!myArea.getText().equals("")){



                    String str = new String("do u want to save this document
");//String to be spoken

                    synthesizer.speakPlainText(str, null);



                    int ch=JOptionPane.showConfirmDialog(null,"do u want to save
this document");


            if(ch==0){

            saveFile();

                    }}


                    String str = new String("you have selected open option ");//String
to be spoken
                    synthesizer.speakPlainText(str, null);

                    openFile();


                    }// end of open event



            else if (e.getSource()==save){

                    String str = new String(" you have selected save option");//String
to be spoken
                    synthesizer.speakPlainText(str, null);
                    saveFile();



                    }// end of save event




              }// end of actionperformed method

public void saveFile()
       {



       int returnVal = fc.showSaveDialog(null);

     if (returnVal == JFileChooser.APPROVE_OPTION) {

       File file = fc.getSelectedFile();

       //this is where a real application would open the file.
       try{
         FileWriter fw = new FileWriter(file);

         BufferedWriter bw = new BufferedWriter(fw);

         bw.write(myArea.getText());

         bw.close();


       }
       catch(Exception ex){}
       }
       }// end of savefile method


public void openFile(){

int returnVal = fc.showOpenDialog(null);

           if (returnVal == JFileChooser.APPROVE_OPTION) {
             File file = fc.getSelectedFile();

             //this is where a real application would open the file.
```

```
                    try{
                       FileReader fr = new FileReader(file);

                       BufferedReader br = new BufferedReader(fr);

                       String temp = "";

                       myArea.setText("");

                       while( (temp = br.readLine()) != null ){
                          myArea.append(temp + "\n");
                       }

                       br.close();
                    }
                    catch(Exception e){}


}

}// end of openFile method


public void run(){


while(true){
         mic.removeOldWord();

         while(!mic.byteArrayComplete()){
            try{
               checking.sleep(200);
            }
            catch(Exception e){
               e.printStackTrace();
            }
         }

         mic.newWord();
         short recordedSample[] = mic.returnShortArray();

         String recognizedWord = engine.getWord(recordedSample);

  if(comm==true){
                              if(recognizedWord.equals("save"))
                        {
```

```
                                        String str = new String("you have selected save option
");//String to be spoken
                                        synthesizer.speakPlainText(str, null);


                                              saveFile();
                                              }// end of save





                                        else if(recognizedWord.equals("open"))
                                 {


                                 if(!myArea.getText().equals("")){


                                  String str = new String("do u want to save this document
");//String to be spoken
                                    synthesizer.speakPlainText(str, null);

                                   int ch=JOptionPane.showConfirmDialog(null,"do u want to save
this document");


                                                    if(ch==0){

                                                    saveFile();
                                                          }}

                                              String str = new String("you have selected open
option");//String to be spoken
                                    synthesizer.speakPlainText(str, null);

                                                    openFile();


                                                    }// end of open

                                              else
if(recognizedWord.equals("Mspaint")||recognizedWord.equals("Notepad")||recognizedW
ord.equals("calc"))

                                              {
```

```
try{r.exec(recognizedWord);
}
catch(Exception ew){}
}




else if(recognizedWord.equals("clear")){

String str = new String("your document is cleared
now ");//String to be spoken

synthesizer.speakPlainText(str, null);

myArea.setText("");

}

else if(recognizedWord.equals("selectall")){

myArea.selectAll();
}




else if(recognizedWord.equals("copy")){

myArea.copy();

}

else if(recognizedWord.equals("cut")){


myArea.cut();



}
```

```
                                        else if(recognizedWord.equals("paste")){


                                        myArea.paste();

                                        }


                                             }//end of commands  if



                                        else
                                        myArea.append(recognizedWord + " ");


                                      }


                         }// end of mic function



               public void synthinit(){




                              try{

                              voiceName = new String("kevin16");




                              desc = new SynthesizerModeDesc(
                                   null, // engine name
                                   "general", // mode name
                                   Locale.US, // locale
                                   null, // running
```

```
                null); // voice
                synthesizer = Central.createSynthesizer(desc);

                if (synthesizer == null) {
                System.exit(1);
                }

                synthesizer.allocate();
                synthesizer.resume();


            desc = (SynthesizerModeDesc) synthesizer.getEngineModeDesc();
            voices = desc.getVoices();
            voice = null;

            for (int i = 0; i < voices.length; i++) {
            if (voices[i].getName().equals(voiceName)) {
            voice = voices[i];

            break;
            }
            }
            if (voice == null) {
                System.err.println(
                    "Synthesizer does not have a voice named "
+voiceName+ ".");
                System.exit(1);
                    }
                synthesizer.getSynthesizerProperties().setVoice(voice);


                }


                catch(Exception ee){}


            }// end of synthesizer function




public static void main(String args[]){

        MyGUI a=new MyGUI();
        }// end of main


}//end of class MyGUI
```