VOICE CONTROLLED SYSTEM APPLICATION WITH VOICEPAD

Project report submitted in the fulfilment of the requirement for the degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Tanvi Kaistha (131214) and **Danish Jain (131241)**

Under the supervision of

Mr. Amol Vasudeva

to

Department of Computer Science and Engineering

Jaypee University of Information Technology, Waknaghat,

Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Voice Control System Application with VoicePad"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering**,** Jaypee University of Information Technology, Waknaghat, is an authentic record of my own work carried out over a period from August 2016 to December 2016 under the supervision of **Mr. Amol Vasudeva** (Assistant Professor, Department of Computer Science & Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)	(Student Signature)
Tanvi Kaistha, 131214	Danish Jain, 131241

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Mr. Amol Vasudeva
Assistant Professor
Department of Computer Science & Engineering
Dated:

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the completion of the project would be incomplete without the mention of the people who made it possible.

We would like to take the opportunity to thank and express their deep sense of gratitude to their faculty mentor **Mr. Amol Vasudeva** for providing their valuable guidance at all stages of the study, their advice, constructive suggestions, positive and supportive attitude and continuous encouragement, without which it would have not been possible to complete the project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which they are about to embark.

We are obliged to all their faculty members of JUIT, for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of their project. We hope that they can build upon the experience and knowledge that they have gained and make a remarkable mark in the future.

CONTENTS

List of Abbreviations

S.No	**Abbreviation**	**Full Form**
1.	SR	Speech Recognition
2.	DTW	Dynamic Time Warping
3.	TTS	Text to Speech
4.	STS	Speech to Text
5.	API	Application Programming Interface
6.	JSAPI	Java Speech API
7.	CART	Classification and Regression Tree
8.	ASR	Automatic Speech Recognition
9.	IJARCSSE	International Journal of Advanced Research in Computer Science and Software Engineering
10	GUI	Graphic User Interface

List of Figures

ABSTRACT

Speech recognition technology is one of the engineering technologies growing at a tremendous rate. It has various applications in different areas and also has many benefits. There are a lot of people in world suffering from various disabilities; many are blind, some cannot use their hands efficiently. The speech recognition systems could help such people. They can share information with people by operating computer through voice input. This gives the user an ease of controlling the system with the help of speech which gives an added advantage of quick action delivery, improves productivity, helps disabled people to use computers, etc. This also helps in aiding people to multitask most of the time while doing some work. There are various approaches which have enabled speech recognition and synthesis and that too in various programming languages and platform. In the proposed approach, they have used Java Speech Application Programming Interface and CloudGarden’s TalkingJava along with necessary updates. Various commands are implemented and tested. The VoicePad enables speech synthesis and speech recognition at the same time. For the performance measurement of the proposed approach, Response time is considered. Our implementation results shows improvement in response time. Our application has the ability to run on three operating systems, Windows, Linux and Macintosh.

CHAPTER 1: INTRODUCTION

1.1 Introduction

We can command our system to do various task like open applications, open chrome, send an email, open calculator or chat when we are bored.

Some tools translate our voice into text a lot quicker than we could ever type.
Thus it helps to,

- Improve our productivity
- Have accurate results
- Provide extreme ease

Speech recognition is a technology which enables a laptop to catch the words spoken by a person with use of microphone [1] [2]. These words are then identified by a speech recognizer, and at last, system will give output of the recognized words. The procedure of speech recognition contains various steps, discussed below one by one.

An ideal scenario within the procedure of speech recognition is that, a speech recognition engine acknowledges all words expressed by a person, basically the performance of a speech recognition engine depends on various 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].

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

In 1940s and 1950s is the foundational period of the technology of speech recognition. In these years work ran on the foundational paradigms of the speech recognition that represents automation and information theoretic models [15]. In the 1960's small vocabularies (order of 10-100 words) of isolated words were recognized, created on simple acoustic-phonetic speech sounds property[3]. Technologies developed were filter banks and time normalization methods [15].

In 1990s the main technologies developed were the methods for statistical learning of acoustic and language models, stochastic language understanding and the methods for implementation of large vocabulary speech understanding systems.
After the five decades of research marketplace has finally encountered the speech recognition technology, benefiting in various ways. The challenge of designing a machine that functions like a smart human is still a major task going forward.

Speech recognition technology is one of the engineering technologies growing at a tremendous rate. It has various applications in different areas and also has many benefits. There are a lot of people in world suffering from various disabilities; many are blind, some cannot use their hands efficiently.

The speech recognition systems could help such people. They can share information with people by operating computer through voice input. This gives the user an ease of controlling the system with the help of speech which gives an added advantage of quick action delivery, improves productivity, helps disabled people to use computers, etc. This also helps in aiding people to multitask most of the time while doing some work.

There are various approaches which have enabled speech recognition and synthesis and that too in various programming languages and platform. In the proposed approach, they have used Java Speech Application Programming Interface and CloudGarden's TalkingJava along with necessary updates. Various commands are implemented and tested.

The VoicePad enables speech synthesis and speech recognition at the same time. For the performance measurement of the proposed approach, Response time is considered.

Our implementation results shows improvement in response time. Our application has the ability to run on three operating systems, Windows, Linux and Macintosh.

1.1.1 Types of speech recognition

We can divide Speech recognition systems into various classes based on their capability of recognizing the words and list of words they consist of. Some classes of speech recognition are:

1.1.1.1 Isolated Speech

Isolated words normally encounter a break between two utterances. It requires one utterance at a time but it can accept more than a single word [4].

1.1.1.2 Connected Speech

Connected words or speech is same as isolated speech but separate utterances with minimum pause between them is allowed.

1.1.1.3 Continuous speech

In Continuous speech the user is allowed to speak naturally. It is also known as computer dictation.

1.1.1.4 Spontaneous Speech

At a fundamental level, Spontaneous speech can be thought of as speech which is natural sounding and not rehearsed. An ASR system with this ability must handle different features of natural speech such as words being run together, for instance "um" and "ah", and even slight impediments.

1.2 Problem Statement

Besides the advantages, hundred per cent perfect speech recognition system is not easy to develop. Various factors lead to decrease in accuracy and performance of the program. Speech recognition process is a tough task for a machine to accomplish, in comparison to a human mind speech recognition programs seems less intelligent.

This is due to a human mind is a gift of god and the capability of thinking, understanding and reacting is usual, while for a computer program it is a difficult task. Firstly, it needs to understand the spoken words and the meanings, and it has to create an apt balance between the words, noise and spaces. A human is capable enough to filter the noise from a speech and on the other hand a machine requires training. A computer needs to be trained to separate the speech sound from the other sounds generated.

Noise factor: the program needs to capture the words said by a human in a distant and clear way. Any additional sound or noise can create interference. Firstly, you should keep system apart from noises and then speak clearly otherwise the machine will face the confusion and will mix up the words.

Some more problems are:

- Hardware problem which means the connection of the microphone with the respective speech engines or other programming aspects of connection.
- There is a noticeable lag time between when we say the word and when it appears on the screen.
- Does not convert audio recordings into text.
- Some applications like Apple’s dictation feature, listens for 30 to 40 seconds as a default character.
- Recognition of different accents is difficult.

1.3 Objectives

- Understanding the speech recognition and its fundamentals.
- Working and application of speech recognition in different areas
- Implementation of speech recognition as a desktop Application
- Developing software that could be used for:
  - Speech Recognition
  - Speech Generation
  - Text Editing
- Tool for operating Machine through voice.

Solution Proposed to the problems stated above in the previous section:

- The updates of JSAPI cause no problem in connecting the hardware and the speech engine.
- Optimised functions aid us to allow the engine to listen to recorded audio.
- Threads if used along with the JSAPI, will remove the problems of lag.
- Use of the Sphinx Engine.

JSAPI Engine Updates:

It configures the speech engine by the use of XML configuration file which is a universally excepted file because XML files provide us with the interoperability between several programming languages which does not provide any hindrance in interaction with different types of systems working on different operating systems or applications which are built using different programming languages.

Each java object is referenced through this file and thus the code area becomes quite clear which is easy for debugging. It provides speaker independent recognition.

1.3 Methodology

Speech Recognition is an emerging technology because of which many developers are not accustomed with this technology. While it only takes a few moments to understand the basic functions of both speech synthesis and speech recognition (after all, most people learn to speak and listen by age two), there are capabilities which are provided by computerized speech that developers will want to understand and utilize.

Despite such substantial research over the last 40 years in speech technology, there are a few limitations in speech synthesis and speech recognition technologies. Speech technology familiar with natural human-to-human speech communication does not always meet the high expectations of users. Understanding the disadvantages - as well as the advantage - is significant for efficient use of speech input and output in a user interface.

An understanding of the abilities and restrictions is also important for developers in speech recognition in making some decisions like whether an application will be of value from the use of speech input and output.

This provides a speech synthesizer which connects with a voice model and performs the operation of text to speech after making a connection with the hardware (speakers) of the system with the help of Java Speech API's speech engine.

The speech synthesis is an automatic process by the engine provided by the JSAPI in which the synthesiser takes in the text as the input, tokenises the text to each words (which is known as pre-processing), and the words are converted to phenomes. The phenomes are the pronunciation of letters and words like letters to the engish grammer. These phenomes are recognised by the speech engine and it converts these phenomes into sound as output through the speakers.

This also provides a speech recognizer which listens to the words and converts this speech to text using a dictionary for recognition. The speech recognition is also an automatic process by the engine by the JSAPI in which the recogniser takes the speech as the input, matches the pattern of the utterences with its rules, checks the grammar file for the words to be recognised according to the phenomes and produces a text output for the application.

NetBeans:

Netbeans is helpful in providing us the following:

- Profiler – for our performance measurement.
- Debugger – for optimising and removing some logical errors.
- Ant build tool – for making an executable JAR file.
- Javadoc generator – for building the map of packages, classes and functions used in the project.
- IDE – for development of the application.

We have opted Java as our software development programming language because it is scalable, platform independent and easy to use and access. Certain models already exist in Java also but again with many issues as discussed earlier.

Our application is a cross platform application which is able to run on Windows, Linux and Macintosh operating system. This was possible because Java itself is platform independent. The added feature we gave was that we provided the arrays of system commands for all the operations.

To tackle some of the problems, we tested a dummy code with the use of updated engine (as discussed earlier) and Threads. It was believed that the problem of lag is removed with the help of using threads. Also with the use of Sphinx, even recorded material if played can be recognized by the speech engine.

Our software is made in two major steps that too in four phases. The first step being the development of the VoicePad. With the VoicePad we can speak into a microphone and, following successful speech recognition (SR), VoicePad will display the words on the screen as text. All or selected words can also be spoken back in a text-to-speech (TTS) voice, through a voice model.

This comprise of two phases, the first being developing the VoicePad with Text-to-Speech property and the other being an update with the Speech-to-Text property. The second major step which is also the third phase is the development of an application which runs through voice and controls the computer system.

1.5 Organisation

1.5.1 Components of Speech recognition System

1.5.1.1 Voice Input

We use microphone to give input to system while pc sound card creates the equivalent digital representation of received audio [8] [9] [10].

1.5.1.2 Digitization

The procedure of conversion of an analog signal into a digital form is called digitization [8]. It consists os sampling and quantization processes. Sampling is a process of converting a signal which is continuous into discrete. Quantization is a process of approximating a range of continuous values.

1.5.1.3 Acoustic Model

A model which is made by taking recordings of audio , their text transcriptions and by using a software that helps create a statistic representations of sounds which make each word. It is used to recognize speech by a speech recognition engine [8]. Words are broken into phonemes by a acoustic model software [10].

1.5.1.4 Language Model

Many natural processing languages uses language modelling like speech recognition which tries to capture language properties and tries to predict the coming word in speech sequence [8]. Comparison of phonemes and words is done by the software language in its built in dictionary [10].

1.5.3 Uses of Speech Recognition Programs

Speech recognition is basically used for two purposes. First and most important is dictation which is translation of words spoken into text. Second is directing the laptop, which means to create a software that would be capable for user to authorize it and operate various applications with voice [4][11].

Dictating something helps a person to write 140 words per minute or even more if he/she can speak fast. This creates an easy way for combining text and helping people everywhere to compose billions of words using devise in less time than writing it all. This could help them to save their effort as well as time. A good alternative to keyboard could be speech technology. If you can't write or are not willing to write then applications of speech recognition can help you do everything and anything that you could do with your keyboard.

1.5.4 Applications

1.5.4.1 From medical perspective

People suffering from various disabilities can use speech recognition system. It could be beneficial for people who have difficulty in using their hands. Speech recognition is used in deaf telephony, such as voicemail to text.

1.5.4.2 From military perspective

Speech Recognition applications are crucial in Air Force. It can reduce workload of pilot. Beside Air Force, these programs can be trained and used in various applications like battle management etc.

1.5.4.3 From educational perspective

People who are disable and who have problems with thought-to-paper communication(they think about an idea and write it on paper which is different) can benefit from the software. [13].

1.5.4.4 Command and Control

Systems that perform various actions and functions on system are called Command and Control systems. Speak words like "Open Netscape" and "Start a new browser" will just do that.

1.5.4.5 Telephony

There are some systems that allow user to speak a command rather than pressing a button.

1.5.4.6 Medical/Disabilities

There are people who have problem in writing due to specific injuries, muscular dystrophy and so on. For Example, people who cannot hear can use a system that could convert user’s speech to text.

1.5.5 The future of speech recognition.

• Accuracy would be better.
• People will accept Dictation speech recognition.
•“intelligent systems” would use this technology. This will help to guess what a speaker wants to say, rather than what he said. This is because people often misspeak and make mistakes unintentionally.
•Many sound systems like microphone can be designed to adapt to background noises, various environment, with better recognition of extra noise which would be discarded.

CHAPTER 2: LITERATURE REVIEW

2.1

TITLE: Basic Research and Implementation Decisions for Text-to-Speech Synthesis-System.

Prof. DRAGOS BURILEANU, Speech Technology and Signal Processing Laboratory, Faculty of Electronics and Telecommunications Department, "Politehnica" Bucharest University, Romania

Speech synthesis which is also termed as Text-to-Speech is a language-dependent area of speech technology. The paper presented a fundamental research work and the related limitations or implementation issues in development of a complete TTS system that too in Romanian language, emphasizing the language particularities and their influence on improving the language processing stage efficiency. The first section describes the standpoint on TTS synthesis as well as the overall architecture of their TTS system.

The sections formulate several important tasks of the natural language processing stage (input text pre-processing, letter-to-phone conversion, acoustic database preparation) and the design philosophy of the corresponding modules, implementation decisions and evaluation experiments.

A distinct section is devoted to an acoustic-phonetic study that assisted the phone-set selection and acoustic database generation. The paper ended with conclusions and a description of the work that is currently in progress at other levels of the TTS system.
The paper described the present status of the TTS synthesis system developed, presenting basic achievements and also recent improvements made at certain processing levels.

Several tasks of the natural language processing stage, implementation details and experimental studies are discussed. Model in the paper follows the already “classical” structure of a TTS system and consists of a (natural) language processing stage that produces a phonetic transcription for the input as the text and a number of symbolic marks.

Syntax control and a signal processing stage that transforms the information received into speech. The system includes pre-processing and syntactic/ prosodic analysis, which is a unit selection algorithm based on minimum distortion criteria.

The paper emphasized many Romanian language particularities and pointed out the fact that high-quality synthesized speech can be obtained only if one takes into account to a deep extent the language-specific linguistic knowledge. Important work is currently in progress to improve the system performance, especially the naturalness of the generated speech.

e.g Extensive linguistic and rosodic studies are now being conducted in order to replace present (simple) manually-created prosodic rules. The part-of speech analysis and the phrase-level parser will be significantly improved, and a new strategy for stress assignments is being investigated. The present version of the TTS system provides a very useful framework both for fundamental research and for further increasing system performance. Its modular approach and the design philosophy of the constituent modules permit easy modifications based on research advances or the demands of specific applications.

2.2

TITLE: Voice activated command and control with speech recognition over WiFi

Tony Ayres, Brian Nolan: Institute of Technology, Blanchardstown, Ireland

The paper has conducted work for the development of a voice-activated and control framework specifically for the control of remote devices in a ubiquitous computing environment. The research considers three different scenario configurations. A recognition grammar for command and control of the robot has been created and implemented in Java, in part in the recognition engine and in part on the robot. Network communications is primarily WLAN with an element of IR where the robot is concerned.

The speech recognition software used includes Sphinx4, Microsoft SAPI and the Java Speech API. Speech technologies are compared in the paper with their benefits. For each given scenario, paper presented and discussed the implementation challenges encountered and their corresponding solutions, including future plans to create additional grammars to extend the framework's range of devices. The ever growing power of these speech recognition systems, coupled with the provision of wireless and Bluetooth networking capabilities, opens the door for a host of new applications to be developed and for old application paradigms to be applied on new frontiers.

With the evolution of the Java platform and language, Java has become suitable for developing speech applications which can command and control devices. The Java Speech API , provides speech recognition and synthesis tools for Java applications. The Java Speech Mark up Language (JSML) [3], which is a subset of XML ,and the Java Speech Grammar Format (JSGF) [4] offer simple but powerful development structures for speech synthesis development and recognition grammars respectively.

To perform speech recognition, a signal must first be acquired, a digital sample of the signal is then taken and the signal is analysed to extract the features of the speech. Traditionally speech recognition and synthesis systems have been developed using low level languages such as C.

The Java Speech API achieves its goal of platform independence by leaving the implementation of the speech engine to the underlying operating system. However the current version of the Java platform has the necessary tools needed to develop and design a speech recognition engine based on the process.

The Java Speech Application programming interface, which is in use, is used to provide a platform independent speech recognition and synthesis interface for Java applications across different platforms like Windows, Linux or Macintosh. SunMicrosystems supply a standard references for the Java Speech API, but they don’t give the implementation. Sphinx, developed at Carnegie Mellon University, is a set of speech recognition development libraries which can be used to create speech applications.

Currently in development is Sphinx 4, a version of Sphinx written in Java. The source code for Sphinx 4 is available; however after some testing with the engine they failed to achieve any significant results (the code had numerous compile errors and no documentation).

2.3 **TITLE**: Controlling Device through Speech Recognition System

Vishakha Karpe, Shilpa Kabadi, Asmita Mutgekar and Manisha Pokharkar,
Department of Computer Eng., Pune, India

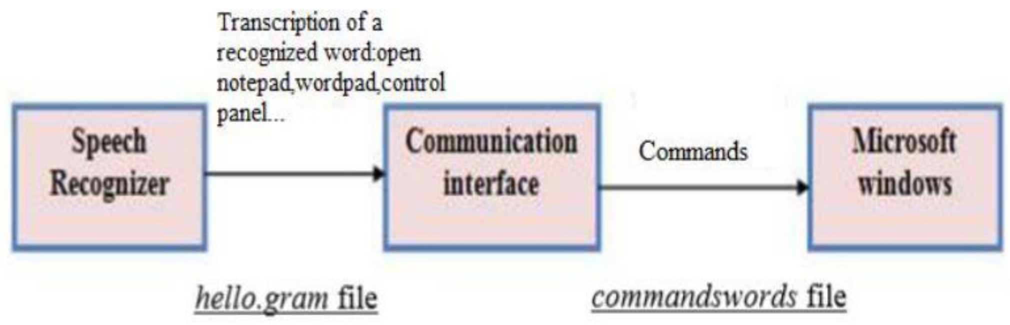


Figure 2.1-Components of Speech Recognition System [11]

In these years, mouse control has become an important part of human computer interaction. Speech Recognition is the process of automatically recognizing a certain word spoken by a particular speaker based on individual information included in speech waves. This technique makes it possible to use the speaker's voice to verify his/her identity and provide controlled access to various services. Speech recognition provides computers with the ability to listen to spoken language and to determine what has been said. Using speech recognition we can give commands to computer and the computer will perform the given task.

The main objective of the project is to construct and develop system to execute commands of operating system by using speech recognition system that is capable of recognizing and responding to speech inputs rather than using traditional means of input (e.g. computer keyboard, mouse), this will lead to save time and reduce effort by the user. This is of great importance to increase the interaction between people and computers by using speech recognition; especially for whom suffer from health problems, for example, persons with disabilities from the movement, this technology helps physically challenged skilled persons.

The application stated helps in reduction in hardware requirement and can be implemented in other electronic devices also. Speech technology is a very popular term right now. It can be divided into two categories: speech recognition and speech synthesis. They have each been in style analysis subjects for over four decades. Speech recognition is very demanded and has several helpful applications.

Voice recognition is employed to map a voice command with its corresponding action. this can be led to by changing speech to text. The program matches the input voice with the voice on that it's trained and maps it to the simplestattainable result. The speech recognition engines area unit accountable for changing physical science signals to digital signals, and so to text. The speech synthesizer engines area unit accountable for changing text to a auditory communication.

This method 1st breaks the words into phonemes, and then converts the phenomes to understandable formate of the engine which is processed and the relevant output is generated. The applications with speech recognition are very helpful for disabled people who have difficulties in typing.

This paper helped us to gather knowledge about different software and hardware materials which are required while developing a speech recognition system. It also helped in understanding various terminologies and system design.

2.4 TITLE: Java Programming Using Voice Input: Adding Java Support to Voice

Christine Masuoka, Dr. Michelle Hugue, Department of Computer Science, University of Maryland, College Park, USA.

They have studied an application named as Voice Code. It is a commercial application implemented with speech recognition and an editor is there to translate speech to code/text. For their project, they added Java support to Voice Code.

Implementation of this Voice Code with Java consisted mainly of commands and logic changes (loop templates, etc.) and their spoken implementation to the Voice Code program. Where possible, they kept the spoken recognized part for Java consistent with voice forms in different languages.

Two major limitations or disadvantages of Voice Code are as follows:

- Difficulty of installation procedure
- Amount of hand use involved in start-up.

They did some tweeks and development in order to reduce these limitations. In order to reduce the typing word, the mouse work or any work that is to be done with the hand, they created batch files for different operations and then these batch files were executed with the help of voice commands.

This is how they reduced most of the hand work. The part for the installation procedure was to be worked upon as they wished to create an installer which would instal the application with only a few mouse clicks.

Paper helped us understand the importance of speech recognition and training of the speech recognition system, which would improve the project development. They helped us in telling that different words with the same meaning can be used differently in order to provide different results. This was seen in the case of a “dot” and a “period”.

Saying “dot” avoids the addition of extra spaces and avoids capitalization of the following word, but saying “period” causes the program to add to extra spaces and to capitalize the first letter of the next word. Presumably, if the user dictated enough code and added enough class and method names to the vocabulary and trained system by reading those into the microphone, the user would not need to explicitly format those symbols, but even then, the user would still need to dictate lots of punctuation, and in the meantime the user would have to explicitly format for capitalization, correct spelling, and manually delete the spaces that system automatically will put between each word.

The paper also posed a reason as to why java should be used while developing speech recognition system. More extensive Java support has the potential to greatly benefit programmers who have difficulty using their hands. Java is the language used for the Computer Science AP exam, and is a fairly common language for introductory computer science classes. For their project, they have designed , developed and added basic Java support to Code.

2.5 **TITLE**: Speech Recognition as Emerging Revolutionary Technology

Parwinder Pal Singh, Bhupinder Singh, Computer Science and Engineering Department, IGCE, PTU Kapurthala

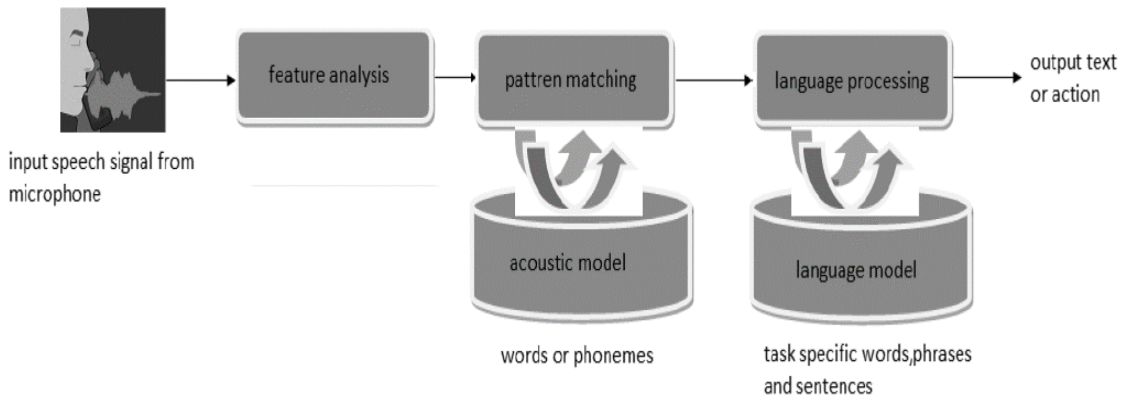


Fig. 2.2-Flowchart of Simple Speech Recognition System [12]

According to them, speech recognition is a new technology in the field of computer science and artificial intelligence. It has changed the way we communicate with the computer and other devices like smart phones. It is a major area of interest for research in this field which is related to artificial intelligence as neural networks and context awareness can be extended to speech recognition and speech synthesiser systems and applications.

From this paper, we got to know that there are mainly three types of speech recognition systems which are as follows:-

- Speaker dependant- A number of voice recognition systems are available on the market. The most powerful can recognize thousands of words. However, they generally require an extended training session during which the computer system becomes accustomed to a particular voice and accent. Such systems are said to be speaker dependent. A speaker dependent system is developed to operate for a single speaker.

- Speaker independent - A speaker independent system is developed to operate for any speaker of a particular type (e.g. American English). These systems are the most difficult to develop, most expensive and accuracy is lower than speaker dependent systems. However, they are more flexible.
- Speaker adaptive - A third variation of speaker models is now emerging, called speaker adaptive. Speaker adaptive systems usually begin with a speaker independent model and adjust these models more closely to each individual during a brief training period.

Then we came to know that the microphone converts the voice signal to an analog signal. This is processed by the sound card of the computer (in our case it is a Realtek sound card for Windows operations), which takes the voice signal to the digital stage. Input from user is also known as voice utterances of the words. This is the binary form of 1s and 0s that make up the machine language stage of computer programming languages. Computers do not hear sounds in any other way. Sound-recognition software has acoustic models (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.

The paper introduces the basics of speech recognition technology and also highlights the difference between different speech recognition systems. In this paper the most common algorithms which are used to do speech recognition are also discussed along with the current and its future use.

CHAPTER 3: SYSTEM DEVELOPMENT

Some of the already existing software of the similar concept, as we all know are Voice Search by Google / Android, Siri by Apple, and Cortana by Windows. But as we have already discussed that there are certain issues with them and our project's sole purpose is to get rid of the maximum issues as we can.

We have opted Java as our software development programming language because it is scalable, platform independent and easy to use and access. Certain models already exist in Java also but again with many issues as discussed earlier.

Our application is a cross platform application which is able to run on Windows, Linux and Macintosh operating system. This was possible because Java itself is platform independent. The added feature we gave was that we provided the arrays of system commands for all the operations.

Java speech models exist with the following steps:

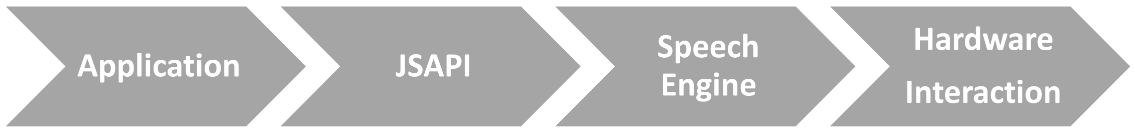


Fig 3.1-Existing Model

To tackle some of the problems, we tested a dummy code with the use of updated engine (as discussed earlier) and Threads. It was believed that the problem of lag is removed with the help of using threads. Also with the use of Sphinx, even recorded material if played can be recognized by the speech engine.

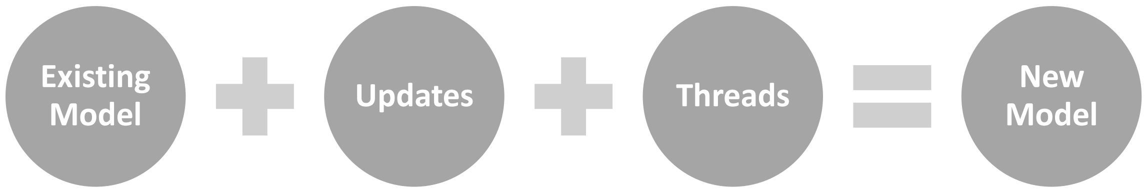


Figure 3.2-Proposed Model

Our software is made in two major steps that too in four phases. The first step being the development of the VoicePad. With the VoicePad we can speak into a microphone and, following successful speech recognition (SR), VoicePad will display the words on the screen as text. All or selected words can also be spoken back in a text-to-speech (TTS) voice, through a voice model.

This comprise of two phases, the first being developing the VoicePad with Text-to-Speech property and the other being an update with the Speech-to-Text property. The second major step which is also the third phase is the development of an application which runs through voice and controls the computer system.

It follows the following steps:

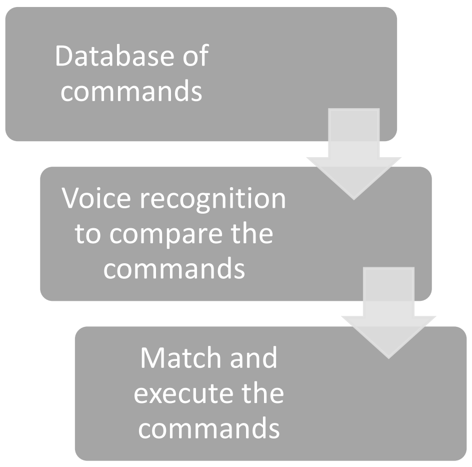

Database of commands
Voice recognition to compare the commands
Match and execute the commands

Figure 3.3-Voice Control System Model

The final phase is the integration of the VoicePad and the Voice Control Application, as well as fixing errors if any.

For our software development process we are using the Iterative Model of Software Development Life Cycle. In Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

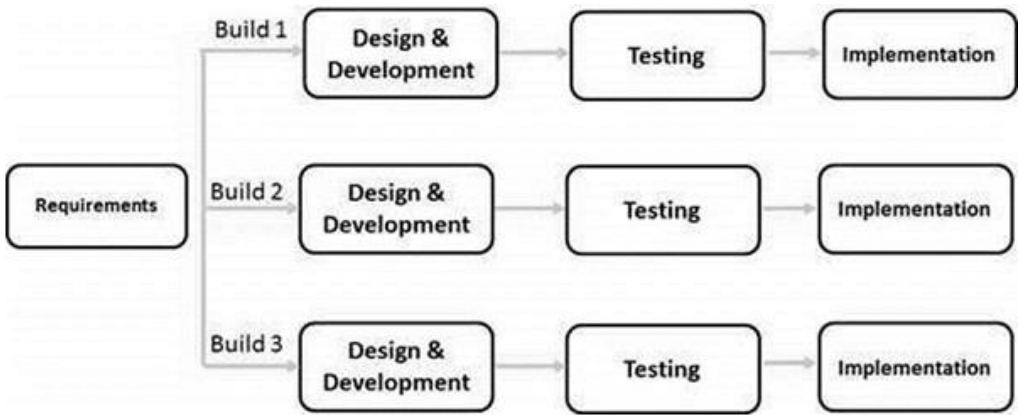


Figure 3.4-Iterative Model

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Requirements:

- 1.6 GHz Processor System
- 128 MB RAM System
- Microphone
- Sound Card
- NetBeans
- Visual Paradigm
- Windows XP or above

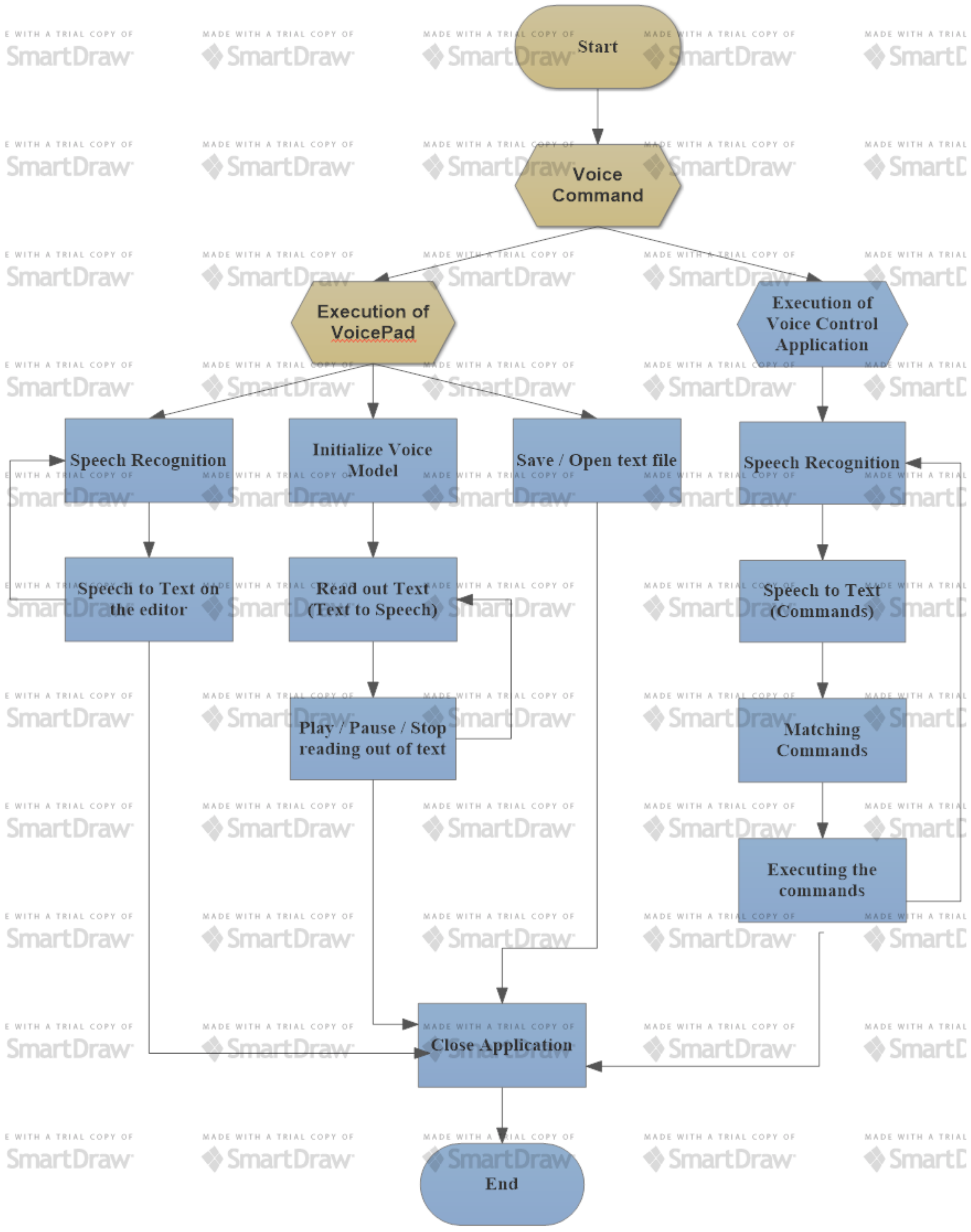


Figure 3.5-Proposed System Architecture

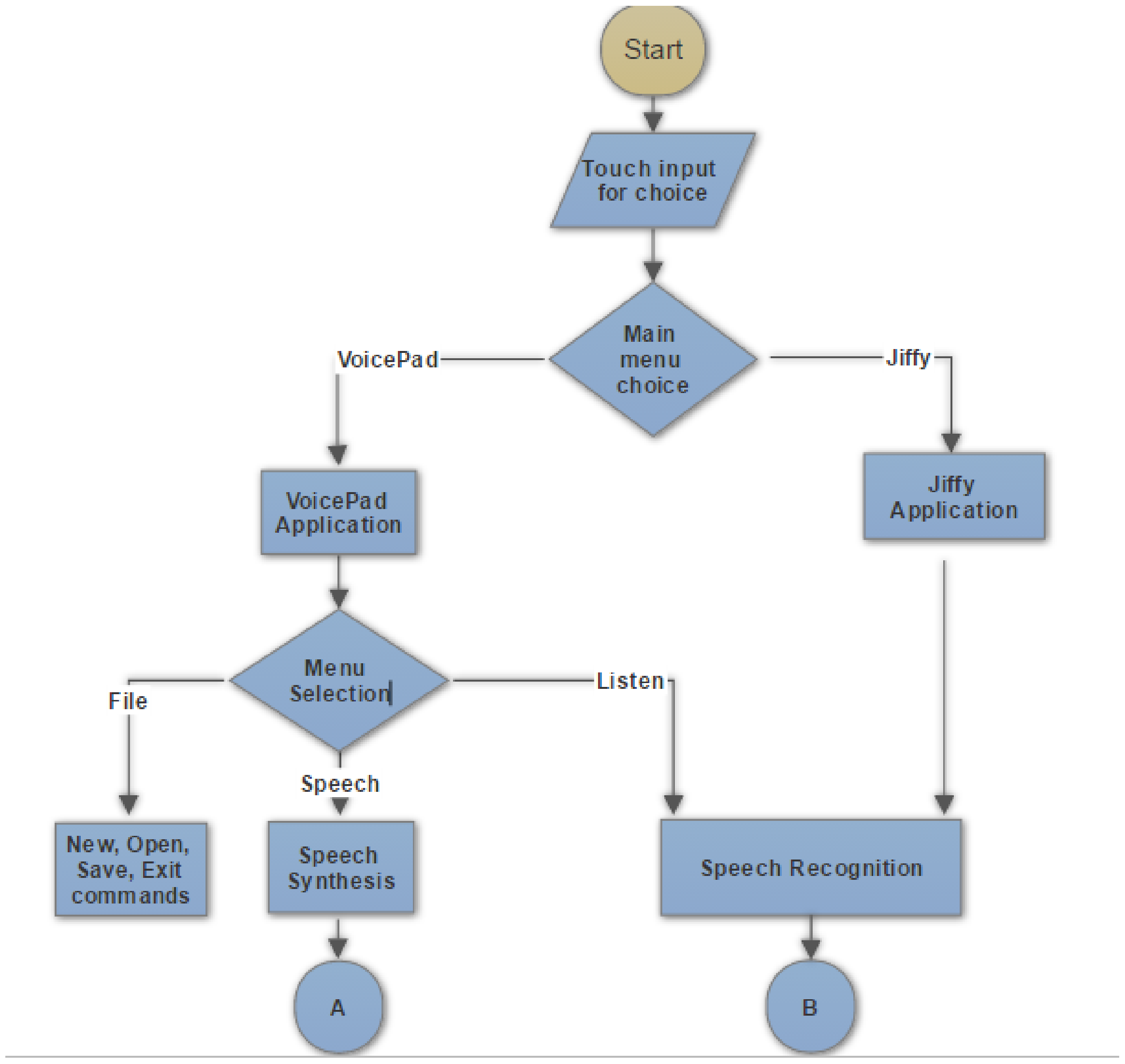


Figure 3.6-Flow Chart (1)

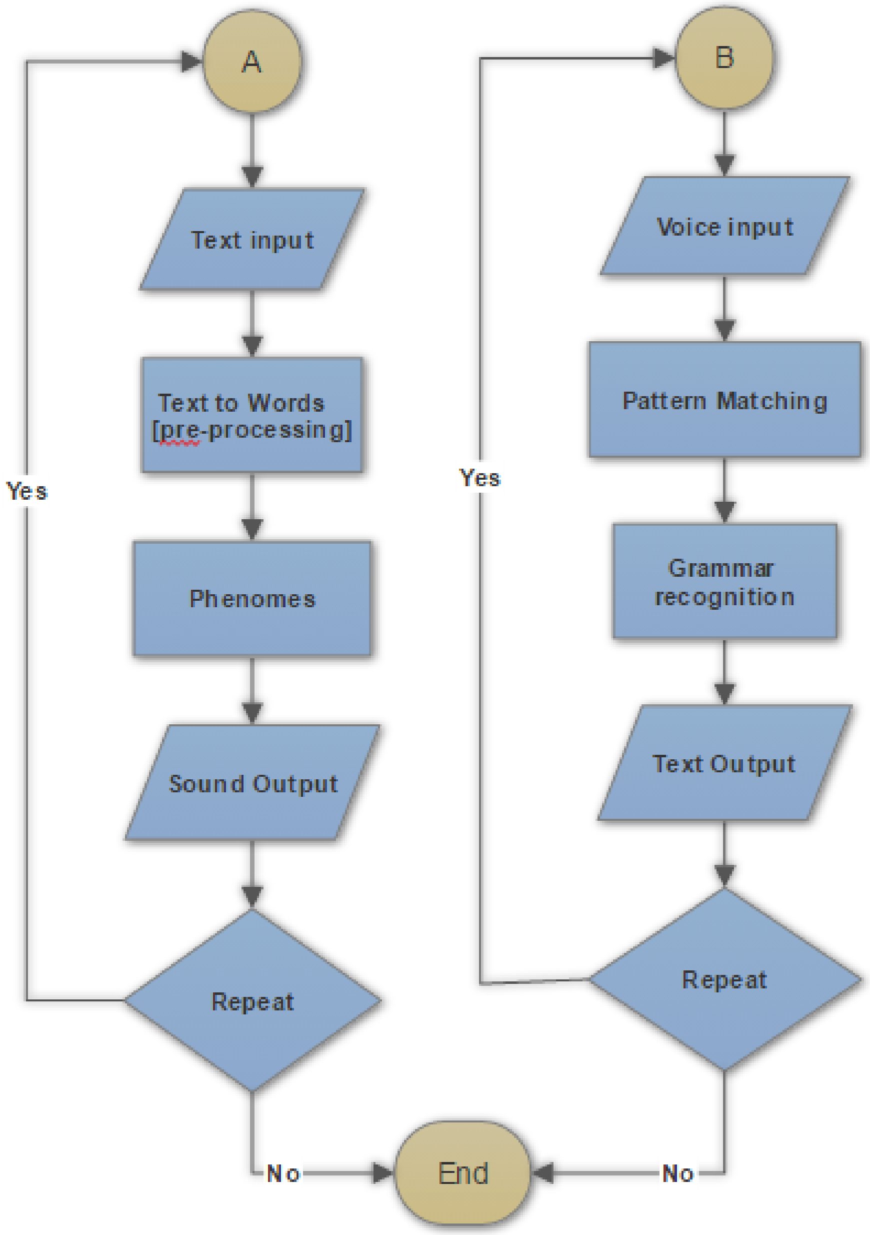


Figure 3.7-Flow Chart (2)

The proposed architecture uses Java Speech Application Programming Interface along with other JAR files.

- A speech-enabled application does not directly interact with the audio hardware of the machine on which it runs.
- The speech engine interacts with the audio hardware.
- The Java Speech API provides a standard and consistent way to access the speech synthesis and speech recognition functionality provided by the speech engine.

This provides a speech synthesizer which connects with a voice model and performs the operation of text to speech after making a connection with the hardware (speakers) of the system with the help of Java Speech API’s speech engine.

The speech synthesis is an automatic process by the engine provided by the JSAPI in which the synthesiser takes in the text as the input, tokenises the text to each words (which is known as pre-processing), and the words are converted to phenomes. The phenomes are the pronunciation of letters and words like letters to the engish grammer. These phenomes are recognised by the speech engine and it converts these phenomes into sound as output through the speakers.

This also provides a speech recognizer which listens to the words and converts this speech to text using a dictionary for recognition. The speech recognition is also an automatic process by the engine by the JSAPI in which the recogniser takes the speech as the input, matches the pattern of the utterences with its rules, checks the grammar file for the words to be recognised according to the phenomes and produces a text output for the application.

Our application development began when we were impressed by the existing systems which enabled us to speak to our devices or applications, and then we were motivated to remove certain problems that are occurring in these applications by developing a more efficient application with less problems. We began our research of speech synthesis and speech recognition with the use of Java programming language and began our development with the iterative process.

At first a general user interface of the VoicePad was developed with the use of Java swing and AWT. Initially it had two menus, a file menu and a speech menu. The file menu has various options like new file option to open a new file to edit, open file option to open an existing text file, save option to save the file which is opened in the editor and finally the exit option to close the VoicePad. The speech menu has options including play, pause, resume and stop options for text-to-speech facility in our application. Later on, when the voice command control application was developed, speech recognition to not interfare in with the speech synthesis, a listen menu was incorporated which has options of hearing and stop hearing the user for speech recognition facility in our application.

Then our VoicePad was integerated with our main application of Jiffy providing two options, one to open the VicePad and other to open the Jiffy voice control application which would only listen to our voices which would be commands and then it would execute those commands which we have spoken.

We include the following JARs:

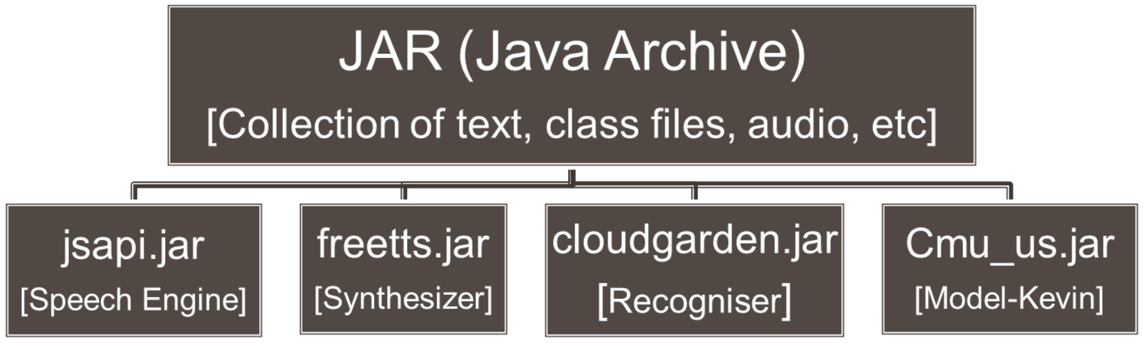


Figure 3.8: JAR Files

In the application, the first class is executed named finalProject which gives us two choices :- one for VoicePad and one for Jiffy. We named our application as Jiffy because of the performance measurement with respect to the response time(which is improved).

Then, if we choose VoicePad, a notepad like window appears which gives us the feature of writing text, listen to the written text, open and save text file and most importantly, speak out to the VoicePad so that the words which we are speaking can be listened by our application and is written out on the text editor.

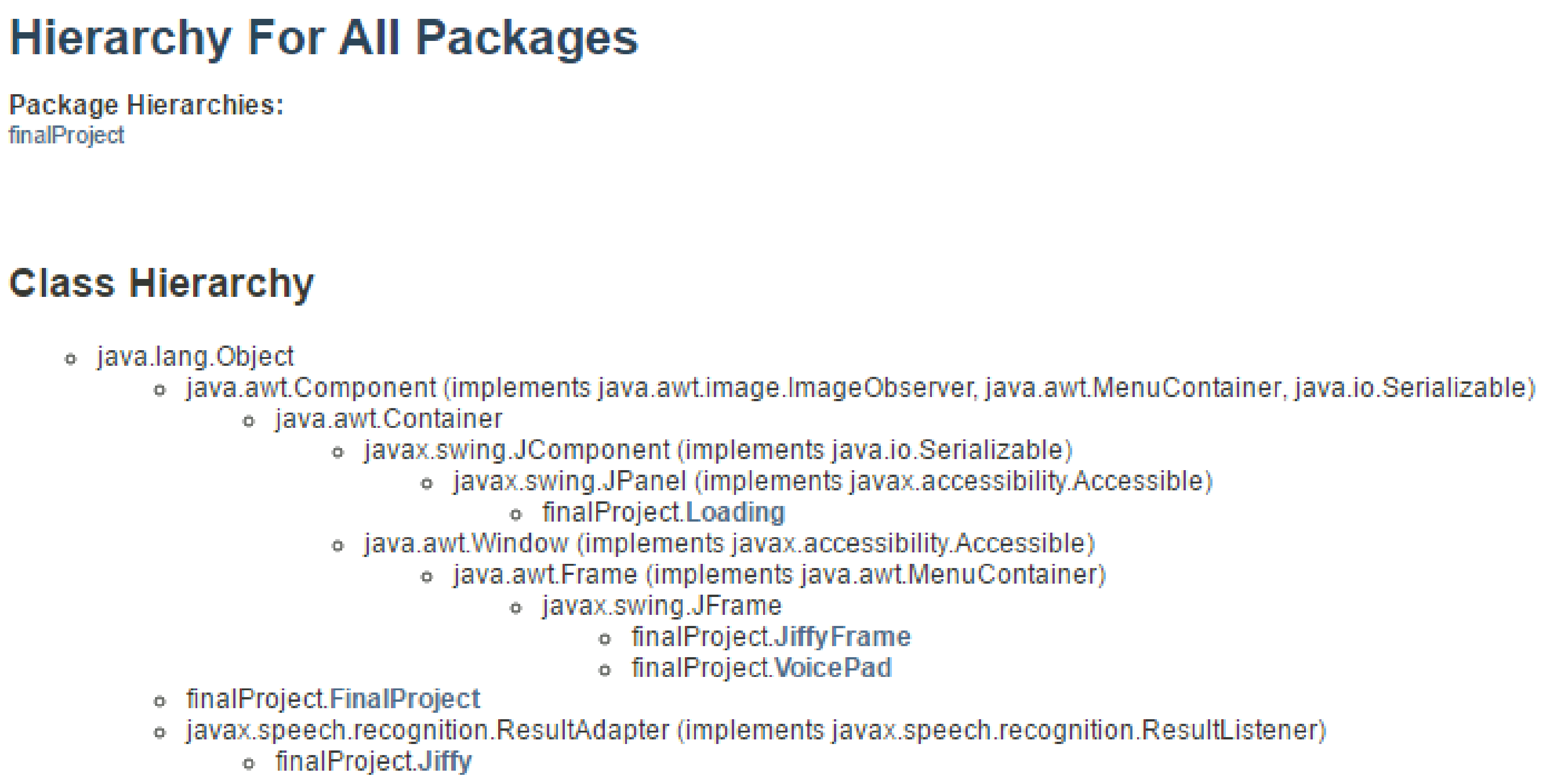
If we choose Jiffy, it executes JiffyFrame and Jiffy classes which opens a window which has the name “Jiffy” on the top, an animation, a status bar (which shows which command we have said to execute), and a list of commands which we can say out loud to our application to execute through speech recognition.

Our Jiffy has the ability to open calculator, open notepad, open paint application, open chrome, open google on chrome and close the application. We can open any application which is installed on the system but we have to put the details of the application in our Jiffy for execution. Once our grammar file has a lot of words, we can even search for anything on google or any other search engine on any browsers installed.

Our application is a cross platform application which is able to run on Windows, Linux and Macintosh operating system. This was possible because Java itself is platform independent. The added feature we gave was that we provided the arrays of system commands for all the operations.

This array was indexed as zero for Linux, one for Windows, and three for Macintosh. Our implemented logic, first detects the operating system, then uses these arrays of system commands to execute a voice command as per the operating system on which the application is running.

We tested our implemented project by taking all the necessary files and making an excutable JAR file for Jiffy project. This executable JAR file was tested on Linux operating system and Macintosh operating system, where our Jiffy was working as per expectations.



Hierarchy For All Packages

Package Hierarchies:
finalProject

Class Hierarchy

- java.lang.Object
    - java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
        - java.awt.Container
            - javax.swing.JComponent (implements java.io.Serializable)
                - javax.swing.JPanel (implements javax.accessibility.Accessible)
                    - finalProject.Loading
            - java.awt.Window (implements javax.accessibility.Accessible)
                - java.awt.Frame (implements java.awt.MenuContainer)
                    - javax.swing.JFrame
                        - finalProject.JiffyFrame
                        - finalProject.VoicePad
    - finalProject.FinalProject
    - javax.speech.recognition.ResultAdapter (implements javax.speech.recognition.ResultListener)
        - finalProject.Jiffy

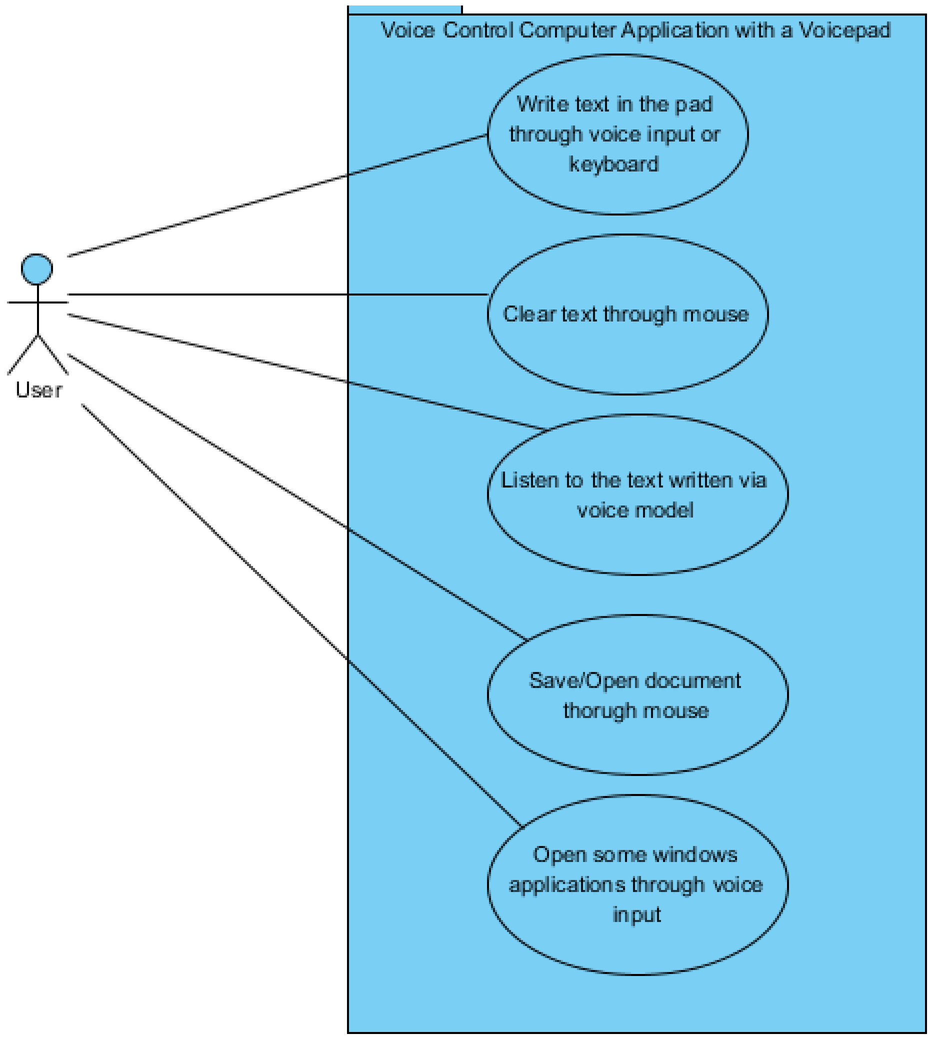

Voice Control Computer Application with a Voicepad
Write text in the pad through voice input or keyboard
Clear text through mouse
Listen to the text written via voice model
Save/Open document thorugh mouse
Open some windows applications through voice input
User

Figure 3.9-Use Case Diagram

Previous phases of the software development:

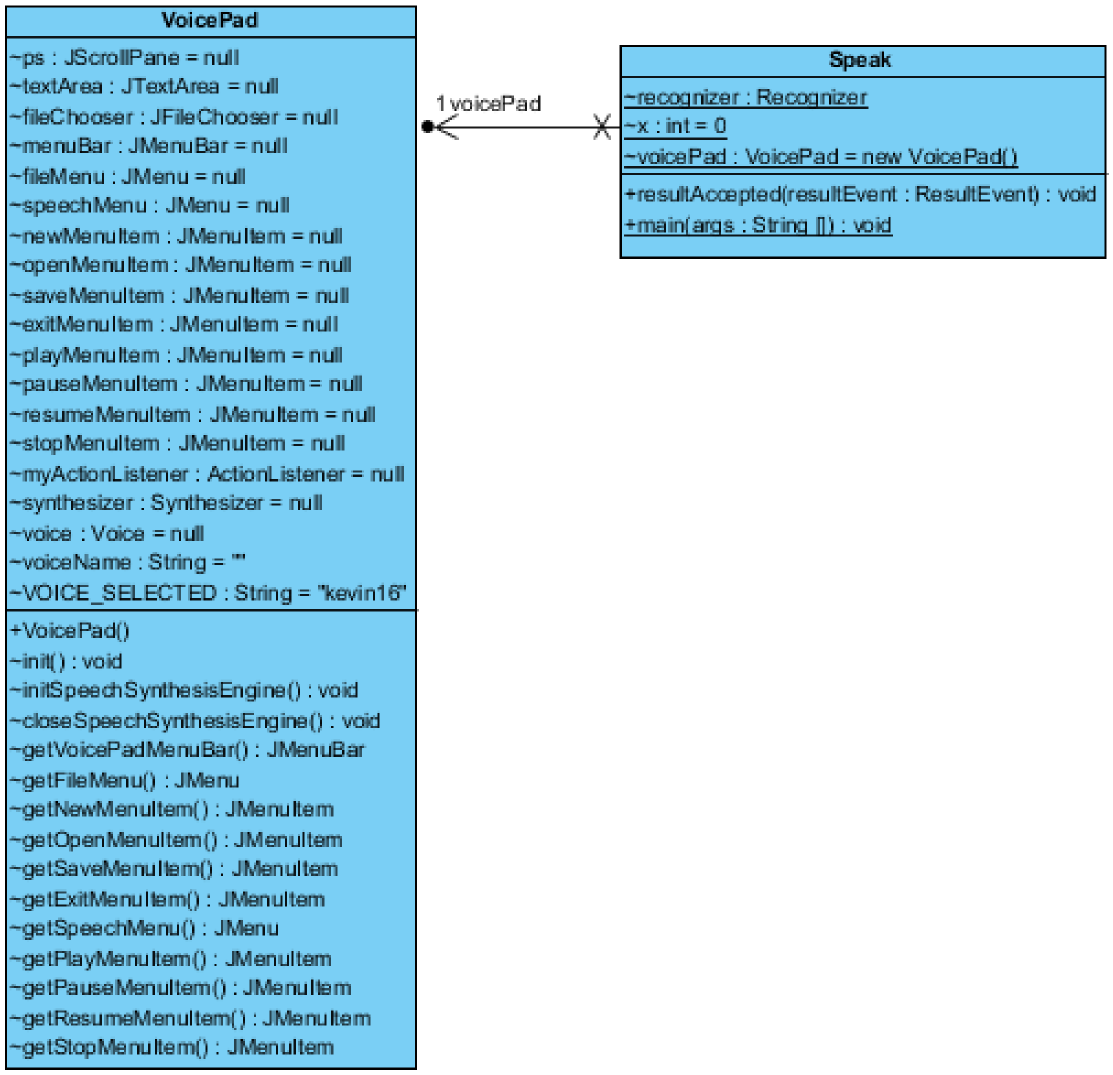

VoicePad
~ps : JScrollPane = null
~textArea : JTextArea = null
~fileChooser : JFileChooser = null
~menuBar : JMenuBar = null
~fileMenu : JMenu = null
~speechMenu : JMenu = null
~newMenuItem : JMenuItem = null
~openMenuItem : JMenuItem = null
~saveMenuItem : JMenuItem = null
~exitMenuItem : JMenuItem = null
~playMenuItem : JMenuItem = null
~pauseMenuItem : JMenuItem = null
~resumeMenuItem : JMenuItem = null
~stopMenuItem : JMenuItem = null
~myActionListener : ActionListener = null
~synthesizer : Synthesizer = null
~voice : Voice = null
~voiceName : String = ""
~VOICE_SELECTED : String = "kevin16"
+VoicePad()
~init() : void
~initSpeechSynthesisEngine() : void
~closeSpeechSynthesisEngine() : void
~getVoicePadMenuBar() : JMenuBar
~getFileMenu() : JMenu
~getNewMenuItem() : JMenuItem
~getOpenMenuItem() : JMenuItem
~getSaveMenuItem() : JMenuItem
~getExitMenuItem() : JMenuItem
~getSpeechMenu() : JMenu
~getPlayMenuItem() : JMenuItem
~getPauseMenuItem() : JMenuItem
~getResumeMenuItem() : JMenuItem
~getStopMenuItem() : JMenuItem
1 voicePad
Speak
~recognizer : Recognizer
~x : int = 0
~voicePad : VoicePad = new VoicePad()
+resultAccepted(resultEvent : ResultEvent) : void
+main(args : String []) : void

Figure 3.10-Class Diagram

Full development of the software:

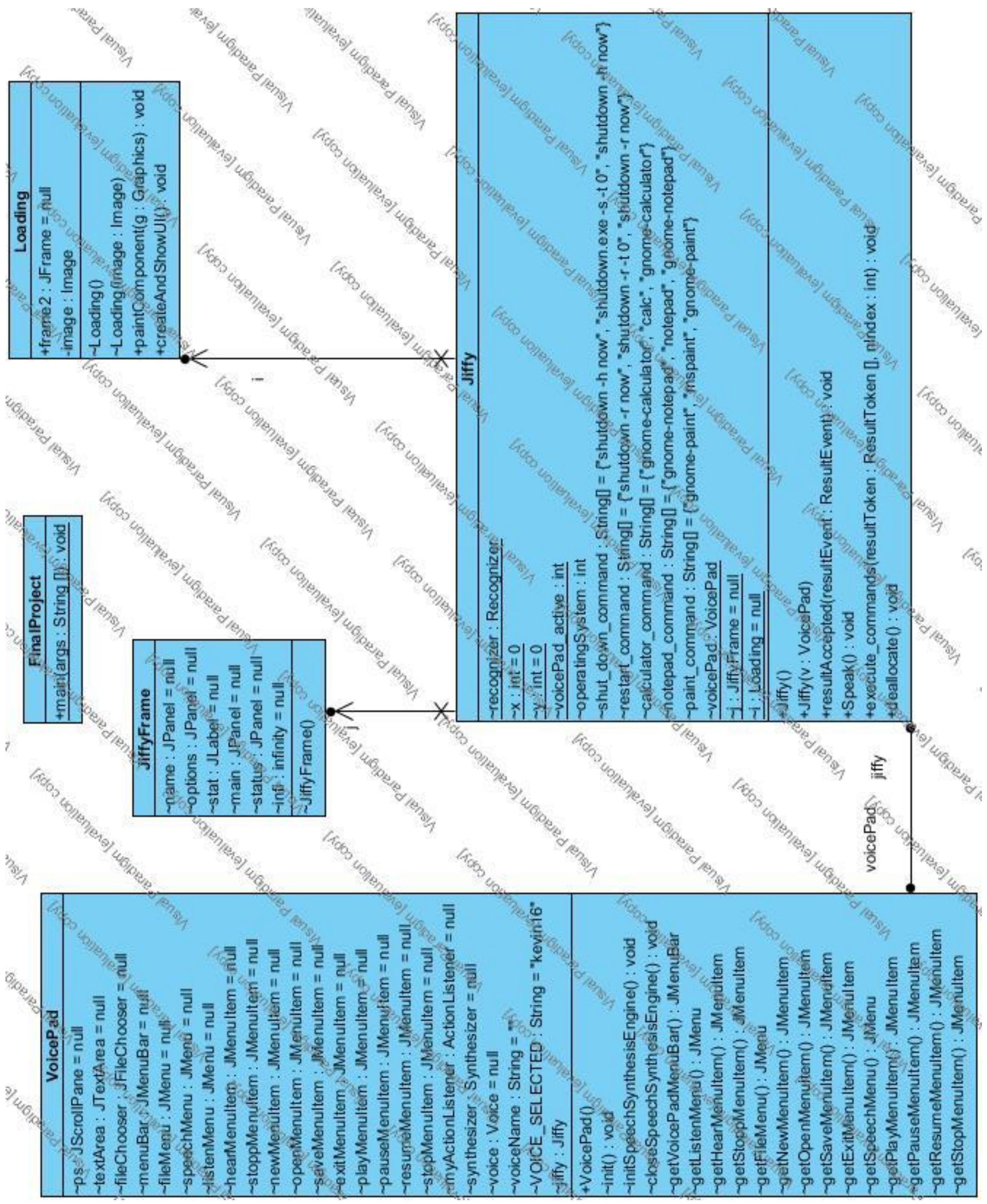

Loading
+frame2 : JFrame = null
-image : Image
~Loading()
~Loading(image : Image)
+paintComponent(g : Graphics) : void
+createAndShowUI() : void
FinalProject
+main(args : String []) : void
JiffyFrame
~name : JPanel = null
~options : JPanel = null
~stat : JLabel = null
~main : JPanel = null
~status : JPanel = null
~inf : infinity = null
~JiffyFrame()
Jiffy
~recognizer : Recognizer
~x : int = 0
~y : int = 0
~voicePad_active : int
~operatingSystem : int
~shut_down_command : String[] = {"shutdown -h now", "shutdown.exe -s -t 0", "shutdown -h now"}
~restart_command : String[] = {"shutdown -r now", "shutdown -r -t 0", "shutdown -r now"}
~calculator_command : String[] = {"gnome-calculator", "calc", "gnome-calculator"}
~notepad_command : String[] = {"gnome-notepad", "notepad", "gnome-notepad"}
~paint_command : String[] = {"gnome-paint", "mspaint", "gnome-paint"}
~voicePad : VoicePad
~j : JiffyFrame = null
~i : Loading = null
~Jiffy()
+Jiffy(v : VoicePad)
+resultAccepted(resultEvent : ResultEvent) : void
+Speak() : void
+execute_commands(resultToken : ResultToken [], nIndex : int) : void
+deallocate() : void
i
j
jiffy
voicePad
VoicePad
~ps : JScrollPane = null
~textArea : JTextArea = null
~fileChooser : JFileChooser = null
~menuBar : JMenuBar = null
~fileMenu : JMenu = null
~speechMenu : JMenu = null
~listenMenu : JMenu = null
~hearMenuItem : JMenuItem = null
~stoppMenuItem : JMenuItem = null
~newMenuItem : JMenuItem = null
~openMenuItem : JMenuItem = null
~saveMenuItem : JMenuItem = null
~exitMenuItem : JMenuItem = null
~playMenuItem : JMenuItem = null
~pauseMenuItem : JMenuItem = null
~resumeMenuItem : JMenuItem = null
~stopMenuItem : JMenuItem = null
~myActionListener : ActionListener = null
~synthesizer : Synthesizer = null
~voice : Voice = null
~voiceName : String = ""
~VOICE_SELECTED : String = "kevin16"
~jiffy : Jiffy
+VoicePad()
~init() : void
~initSpeechSynthesisEngine() : void
~closeSpeechSynthesisEngine() : void
~getVoicePadMenuBar() : JMenuBar
~getListenMenu() : JMenu
~getHearMenuItem() : JMenuItem
~getStoppMenuItem() : JMenuItem
~getFileMenu() : JMenu
~getNewMenuItem() : JMenuItem
~getOpenMenuItem() : JMenuItem
~getSaveMenuItem() : JMenuItem
~getExitMenuItem() : JMenuItem
~getSpeechMenu() : JMenu
~getPlayMenuItem() : JMenuItem
~getPauseMenuItem() : JMenuItem
~getResumeMenuItem() : JMenuItem
~getStopMenuItem() : JMenuItem

Figure 3.11-Class Diagram (Full)

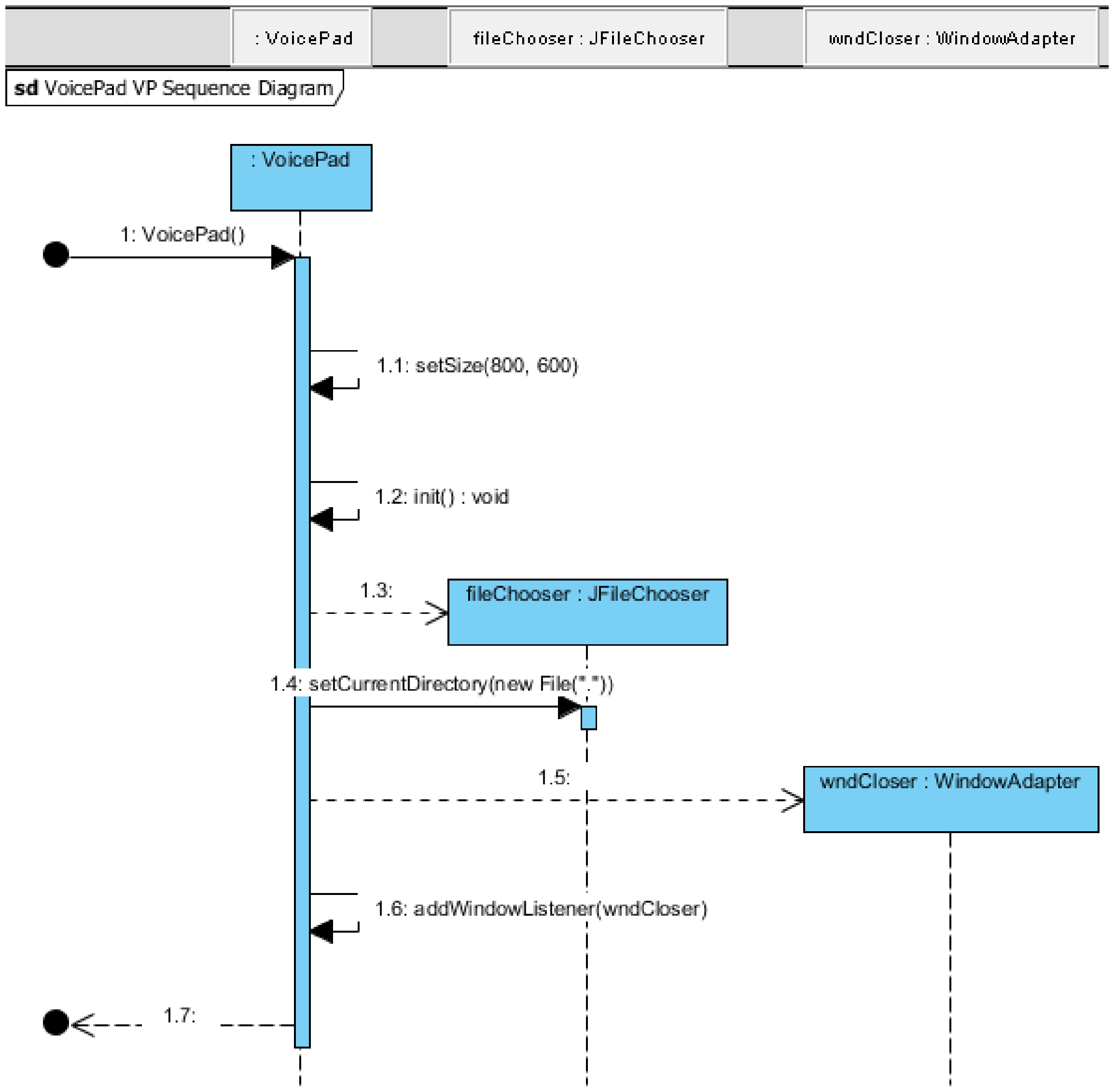

: VoicePad
fileChooser : JFileChooser
wndCloser : WindowAdapter
sd VoicePad VP Sequence Diagram
: VoicePad
1: VoicePad()
1.1: setSize(800, 600)
1.2: init() : void
1.3:
fileChooser : JFileChooser
1.4: setCurrentDirectory(new File("."))
1.5:
wndCloser : WindowAdapter
1.6: addWindowListener(wndCloser)
1.7:

Figure 3.12-VoicePad Sequence Diagram

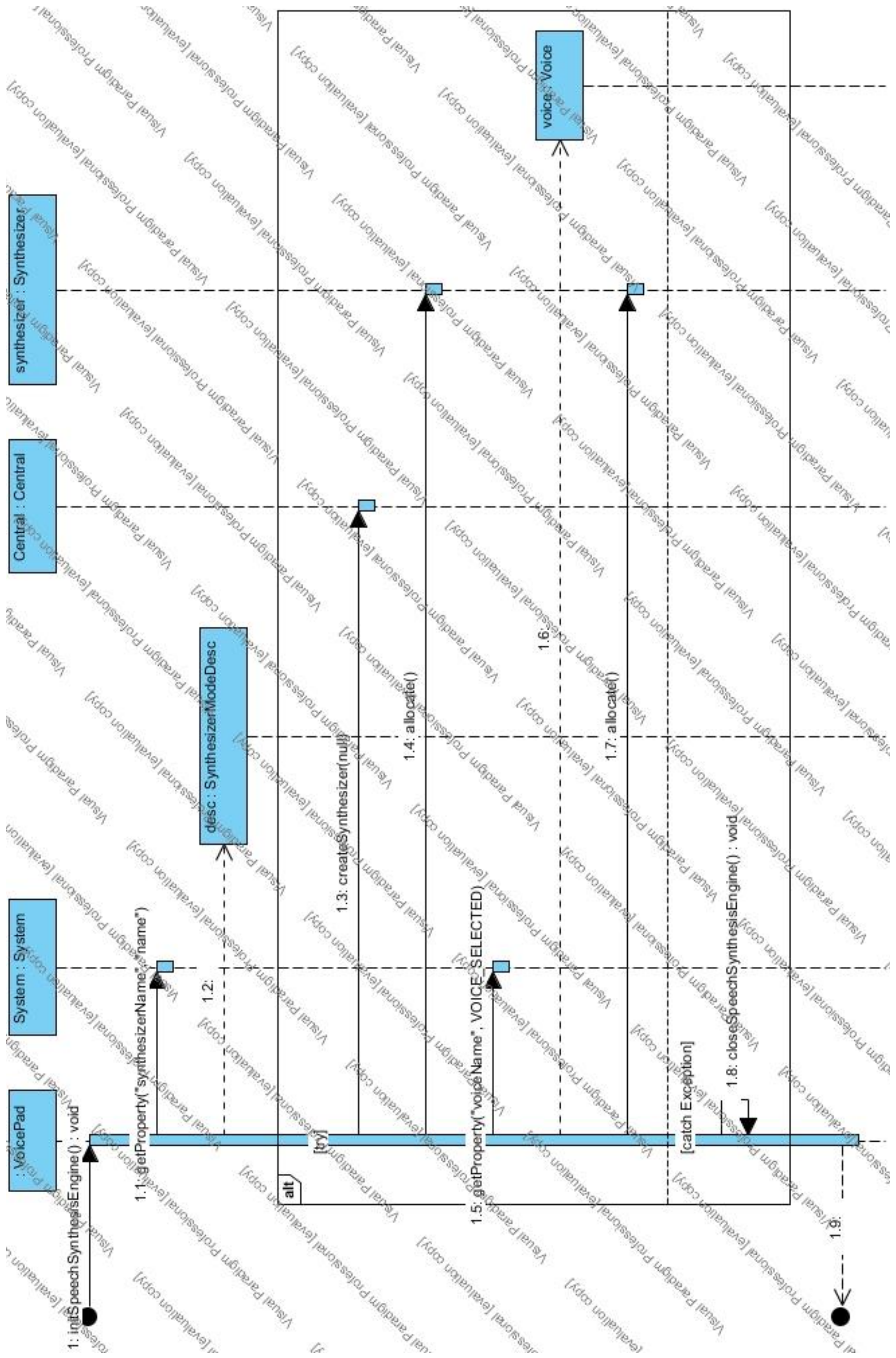

: VoicePad
System : System
desc : SynthesizerModeDesc
Central : Central
synthesizer : Synthesizer
voice : Voice
1: initSpeechSynthesisEngine() : void
1.1: getProperty("synthesizerName", name")
1.2:
alt
[try]
1.3: createSynthesizer(null)
1.4: allocate()
1.5: getProperty("voiceName", VOICE_SELECTED)
1.6:
1.7: allocate()
[catch Exception]
1.8: closeSpeechSynthesisEngine() : void
1.9:

Figure 3.13-Init() Sequence Diagram

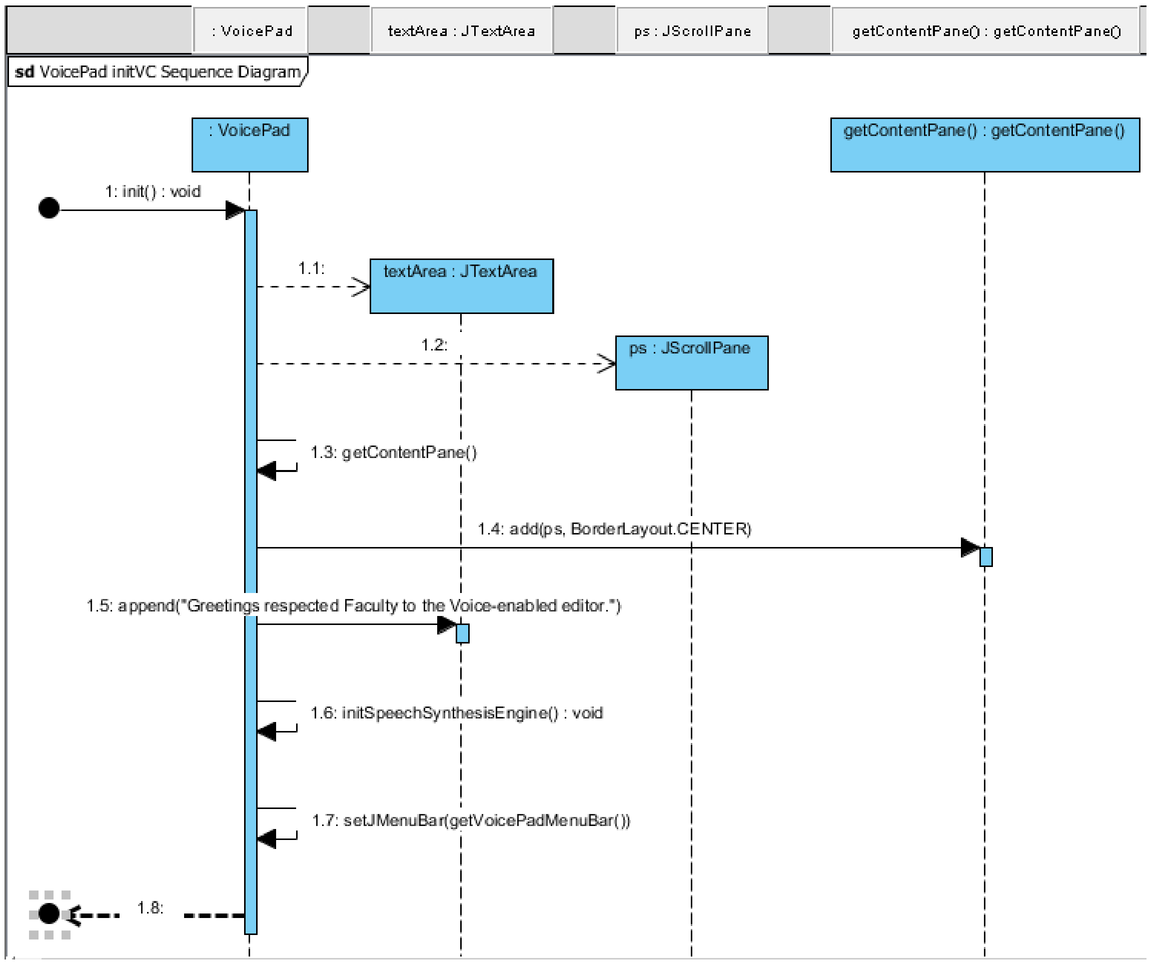

: VoicePad
textArea : JTextArea
ps : JScrollPane
getContentPane() : getContentPane()
sd VoicePad initVC Sequence Diagram
: VoicePad
getContentPane() : getContentPane()
1: init() : void
1.1:
textArea : JTextArea
1.2:
ps : JScrollPane
1.3: getContentPane()
1.4: add(ps, BorderLayout.CENTER)
1.5: append("Greetings respected Faculty to the Voice-enabled editor.")
1.6: initSpeechSynthesisEngine() : void
1.7: setJMenuBar(getVoicePadMenuBar())
1.8:

Figure 3.14-Init Speech Sequence Diagram

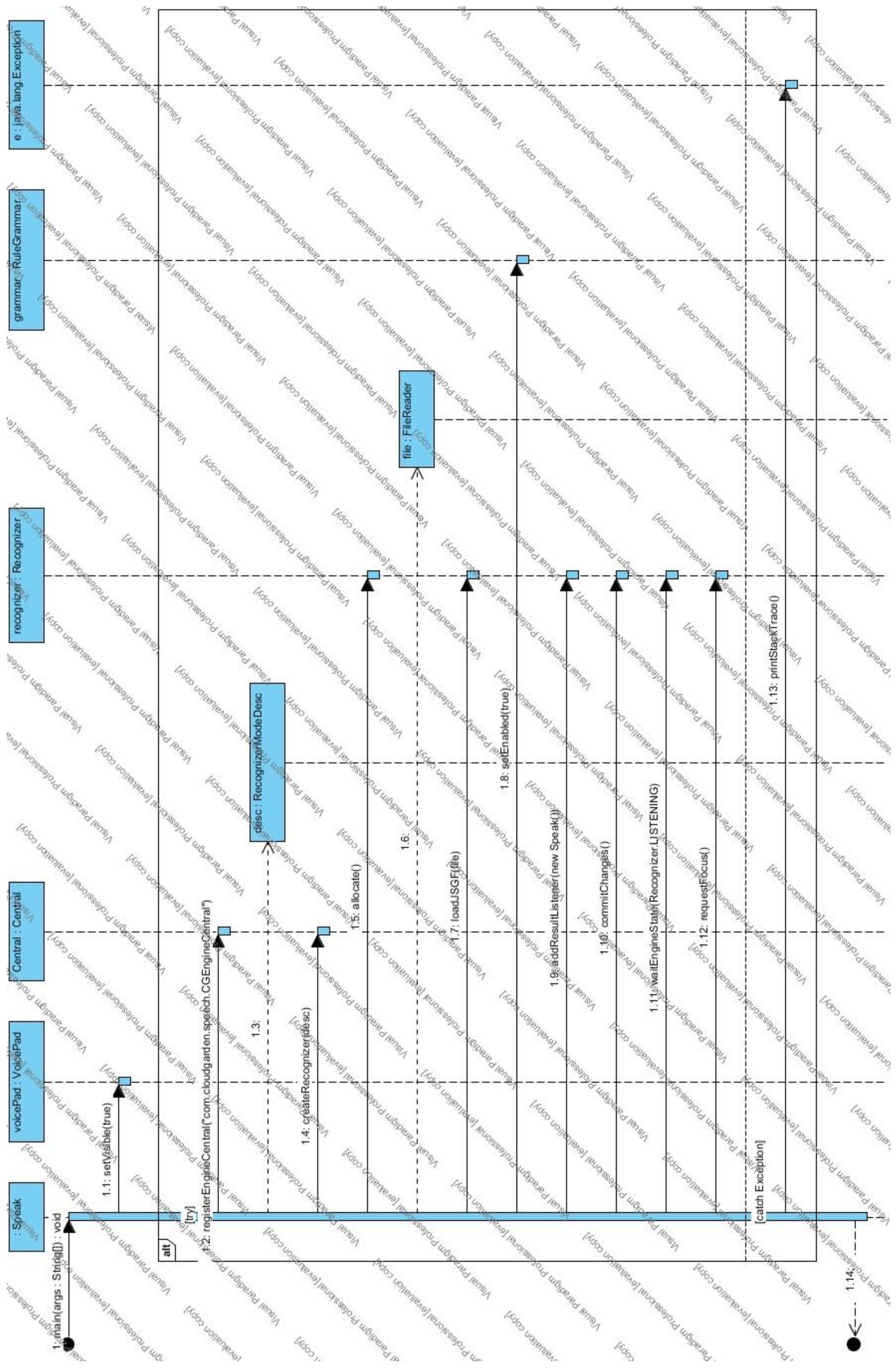


Figure 3.15-main() Sequence Diagram

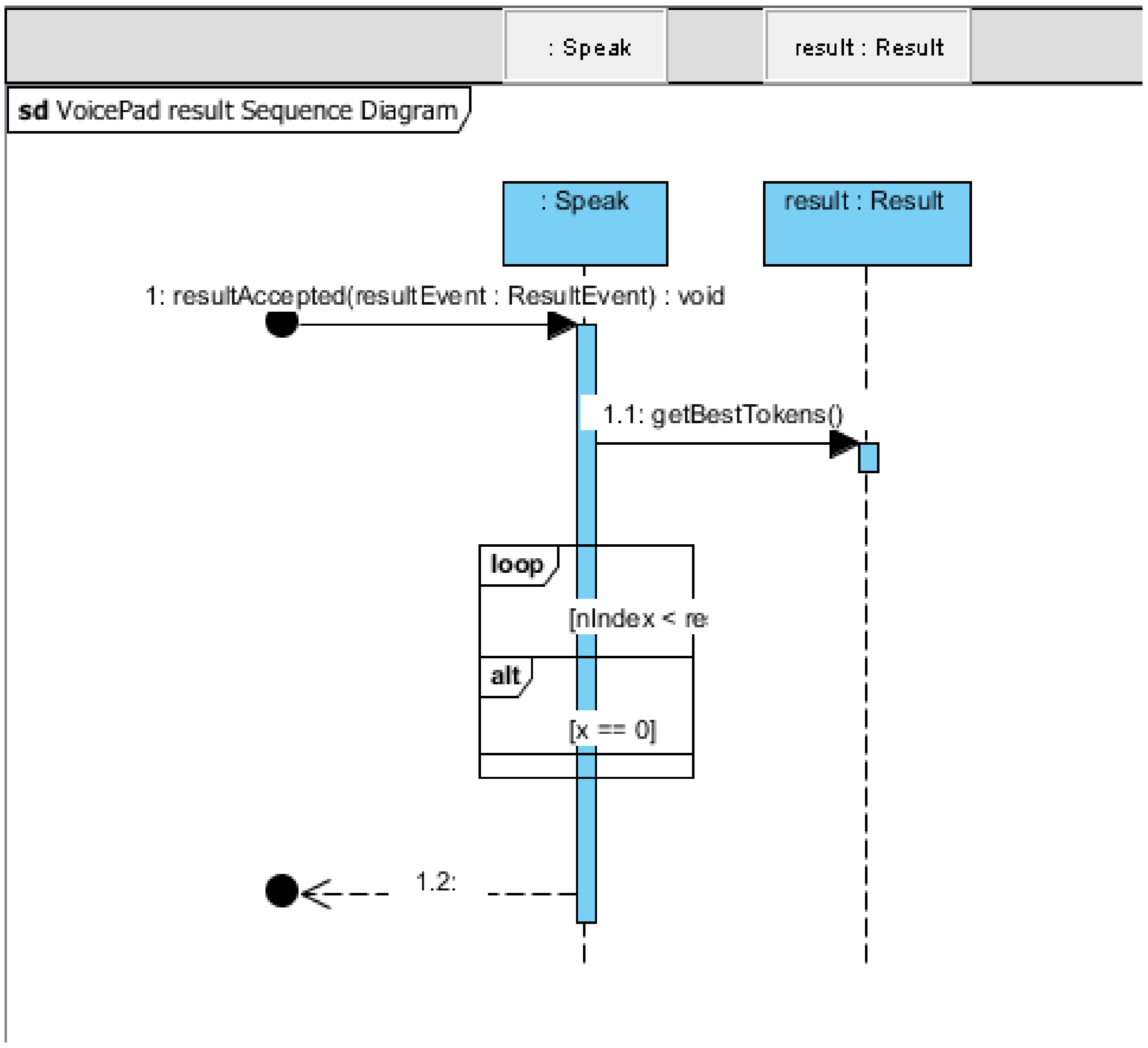


Figure 3.16-ResultEvent() Sequence Diagram

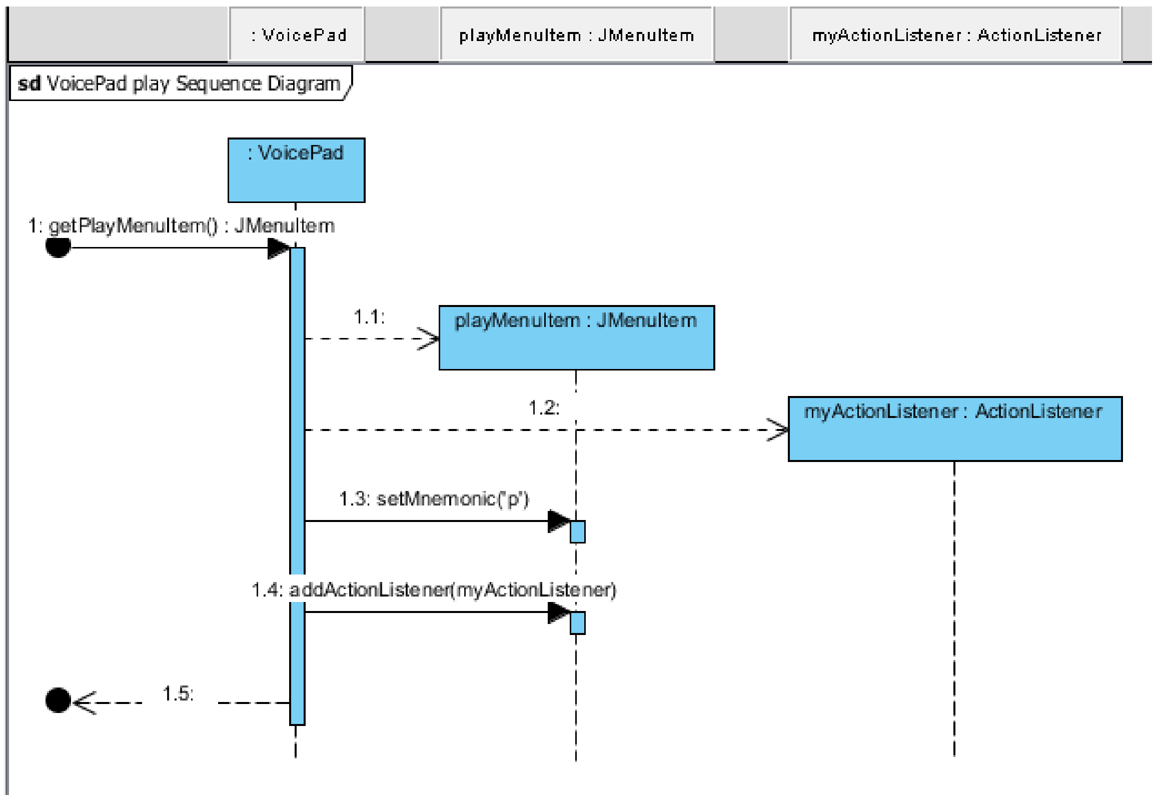
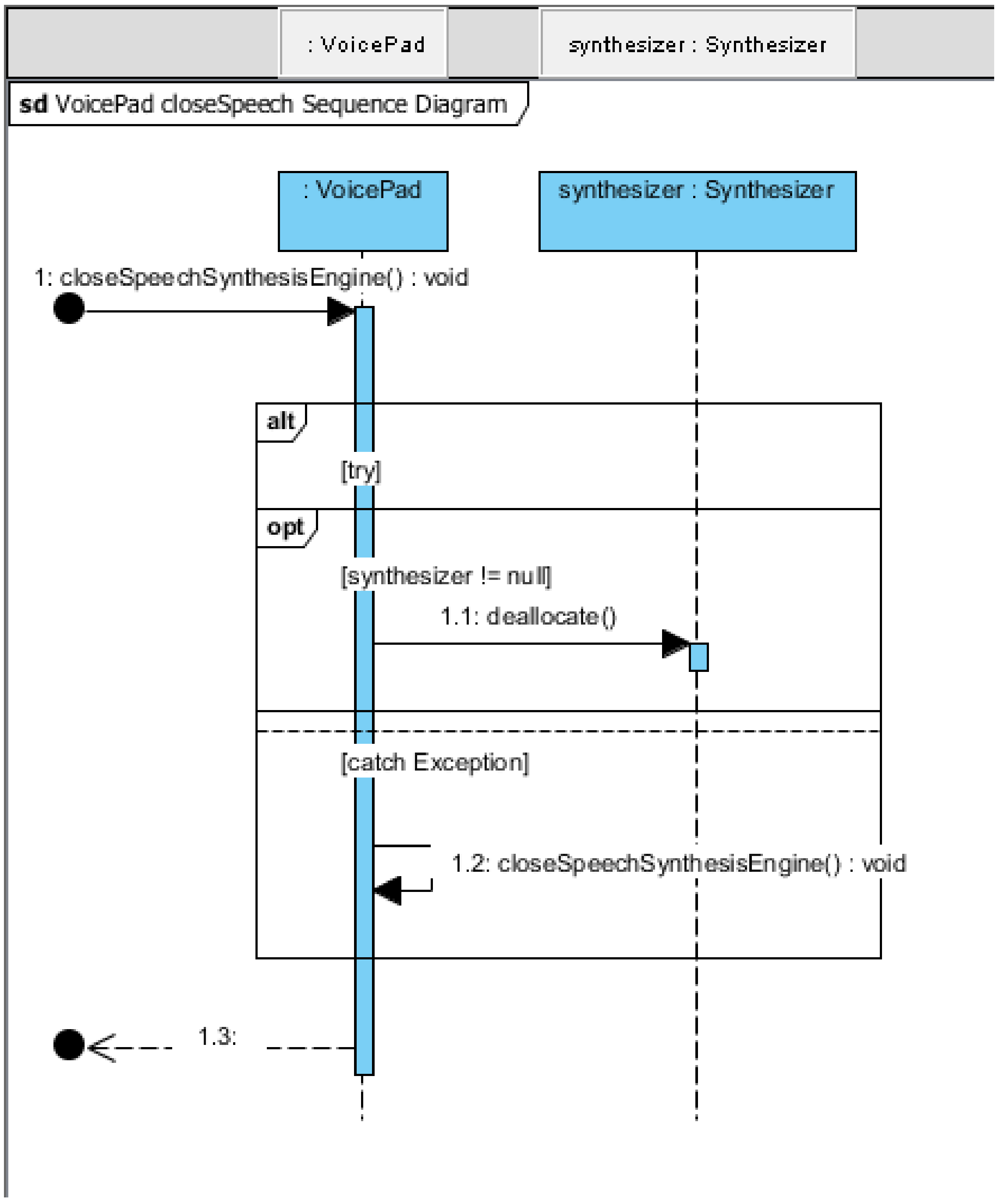

: VoicePad
playMenuItem : JMenuItem
myActionListener : ActionListener
sd VoicePad play Sequence Diagram
: VoicePad
1: getPlayMenuItem() : JMenuItem
1.1:
playMenuItem : JMenuItem
1.2:
myActionListener : ActionListener
1.3: setMnemonic('p')
1.4: addActionListener(myActionListener)
1.5:

Figure 3.17-Play Voice Sequence Diagram



: VoicePad
synthesizer : Synthesizer
sd VoicePad closeSpeech Sequence Diagram
: VoicePad
synthesizer : Synthesizer
1: closeSpeechSynthesisEngine() : void
alt
[try]
opt
[synthesizer != null]
1.1: deallocate()
[catch Exception]
1.2: closeSpeechSynthesisEngine() : void
1.3:

Figure 3.18-CloseSpeech Sequence Diagram

CHAPTER 4: PERFORMANCE ANALYSIS

The performance of the software is measurable by only measuring the efficiency of the speech engine used in the project as well as its related components which are used. One of the simplest and earliest approaches to pattern recognition is the template approach.

Matching is a generic operation in pattern recognition which is used to determine the similarity between two entities of the same type. In template matching the template or prototype of the pattern to be recognized is available.

4.1 Data-Flow Analysis Algorithm

We have used the CloudGarden Java Speech API for Java speech recognition and the FreeTTS for Java speech synthesis. The performance measurement for our application was done through data-flow analysis algorithm which measures the response time for various factors like the flow of data , execution time and allocation time.

Profiling is achieved by instrumenting either the program source code or its binary executable form using a tool called a profiler (or code profiler). Profilers may use a number of different techniques, such as event-based, statistical, instrumented, and simulation methods.

And luckily, NetBeans has an inbuilt profiler attached to it which works effectively in analyzing the project being worked on.

NetBeans profiler is a fully featured Java profiling tool integrated into the NetBeans IDE. The features include CPU, memory, threads, locks and SQL queries profiling as well as basic JVM monitoring, allowing developers to be more productive in solving and analysing performance and memory issues.

Now, to measure the data flow, time of execution, threads allocation, indexing of arrays, and overall performance, we apply the Data-Flow Analysis Algorithm.
A data-flow algorithm gathers information about the definition and use of data in a procedure or a set of programs. The algorithm is usually applied to some intermediate representation of a program. In this, we have a single entry control flow and each part affects the data nad execution time of the program. Data-flow algorithm gather this local information and infer global data flow from it.

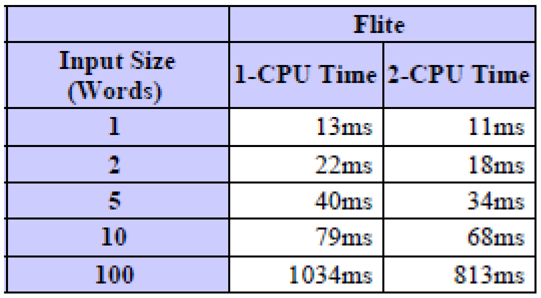
The information is derived by the help of Profiling.
Profiling is an important resource in the empirical analysis of an algorithms running time. Measuring time spent on different segments of a program can pinpoint a bottleneck in the program's performance that can be missed by an abstract deliberation about the algorithm's basic operations. The process of getting such data is called profiling. In simple words In software engineering, profiling ("program profiling", "software profiling") is a form of dynamic program analysis that measures, for example, the space (memory) or time complexity of a program, the usage of particular instructions, or the frequency and duration of function calls. Most commonly, profiling information serves to aid program optimization and performance analysis.

It tells us:

- **Telemetry** - displaying actual CPU utilization and GC overhead, heap size and usage, surviving generations and GC intervals, numbers of threads and loaded classes.
- **Methods** - displaying call trees and hot methods, execution times and optionally invocation numbers.
- **Objects** - displaying live or all allocated objects with numbers of instances and total size, optionally showing allocation call trees.
- **Threads** - displaying process threads, times and states in a timeline.

Comparing existing model and system:

As we can see, by the study of Willie Walker, Paul Lamere, and Philip Kwok[19], the time taken by the system is as shown in the table by them.



	Flite	
Input Size (Words)	1-CPU Time	2-CPU Time
1	13ms	11ms
2	22ms	18ms
5	40ms	34ms
10	79ms	68ms
100	1034ms	813ms

Figure 4.1-CPU time for Flite [19]

Now the main agenda is what we have changed. We have incorporated freeTTS developed by Sun microsystems along with the cloudgarden speech api. Cloudgarden speech api is only for speech recognition purposes that we have implemented in the project.

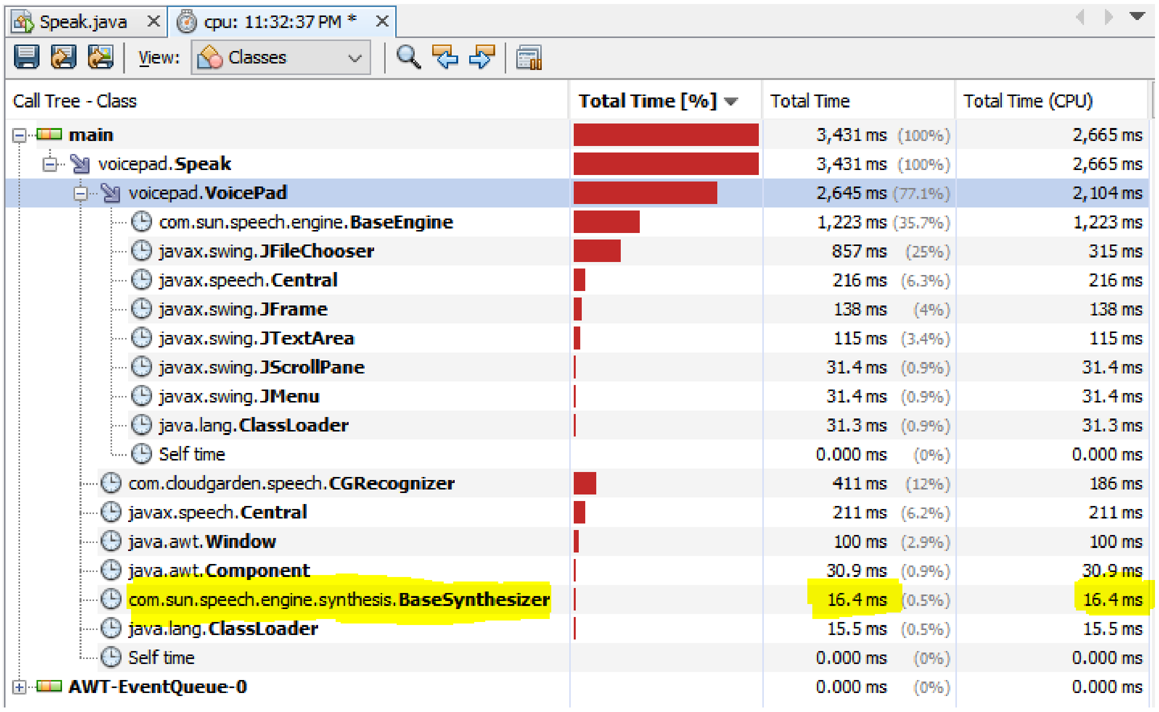
We were lucky enough to find that the process of Profiling is an added inbuilt feature in NetBeans which we were using for our application development. Netbeans was also helpful in providing us the following:

- Profiler – for our performance measurement.
- Debugger – for optimising and removing some logical errors.
- Ant build tool – for making an executable JAR file.
- Javadoc generator – for building the map of packages, classes and functions used in the project.

We did the profiling through NetBeans on our project with the welcome words as the string:

"Greetings respected Faculty to the Voice-enabled editor."

And then we closed the application and checked the result. And we measured the execution of the speaking of the text to be only 16.5 ms which is even less than two words spoken in the above comparison.



Speak.java | cpu: 11:32:37 PM *

View: Classes

Call Tree - Class	Total Time [%]	Total Time	Total Time (CPU)
main		3,431 ms (100%)	2,665 ms
voicepad.**Speak**		3,431 ms (100%)	2,665 ms
voicepad.**VoicePad**		2,645 ms (77.1%)	2,104 ms
com.sun.speech.engine.**BaseEngine**		1,223 ms (35.7%)	1,223 ms
javax.swing.**JFileChooser**		857 ms (25%)	315 ms
javax.speech.**Central**		216 ms (6.3%)	216 ms
javax.swing.**JFrame**		138 ms (4%)	138 ms
javax.swing.**JTextArea**		115 ms (3.4%)	115 ms
javax.swing.**JScrollPane**		31.4 ms (0.9%)	31.4 ms
javax.swing.**JMenu**		31.4 ms (0.9%)	31.4 ms
java.lang.**ClassLoader**		31.3 ms (0.9%)	31.3 ms
Self time		0.000 ms (0%)	0.000 ms
com.cloudgarden.speech.**CGRecognizer**		411 ms (12%)	186 ms
javax.speech.**Central**		211 ms (6.2%)	211 ms
java.awt.**Window**		100 ms (2.9%)	100 ms
java.awt.**Component**		30.9 ms (0.9%)	30.9 ms
com.sun.speech.engine.synthesis.**BaseSynthesizer**		16.4 ms (0.5%)	16.4 ms
java.lang.**ClassLoader**		15.5 ms (0.5%)	15.5 ms
Self time		0.000 ms (0%)	0.000 ms
AWT-EventQueue-0		0.000 ms (0%)	0.000 ms

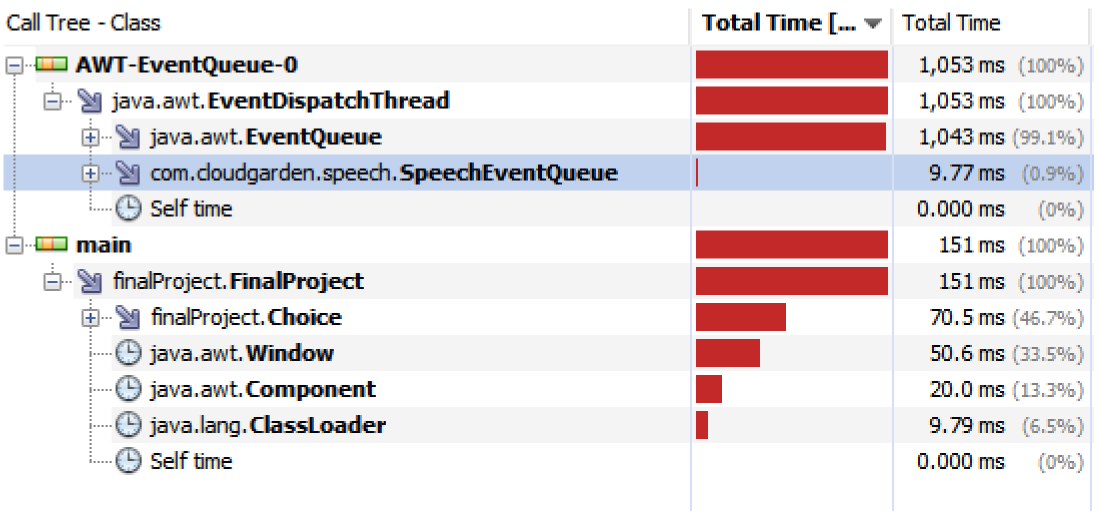
Figure 4.2-CPU time for the system developed

And the whole execution including speech recognition, GUI by swings and AWT and Text-to-Speech was completed in merely 3 seconds.

It is clearly seen that the performance of our system is beyond comparison.

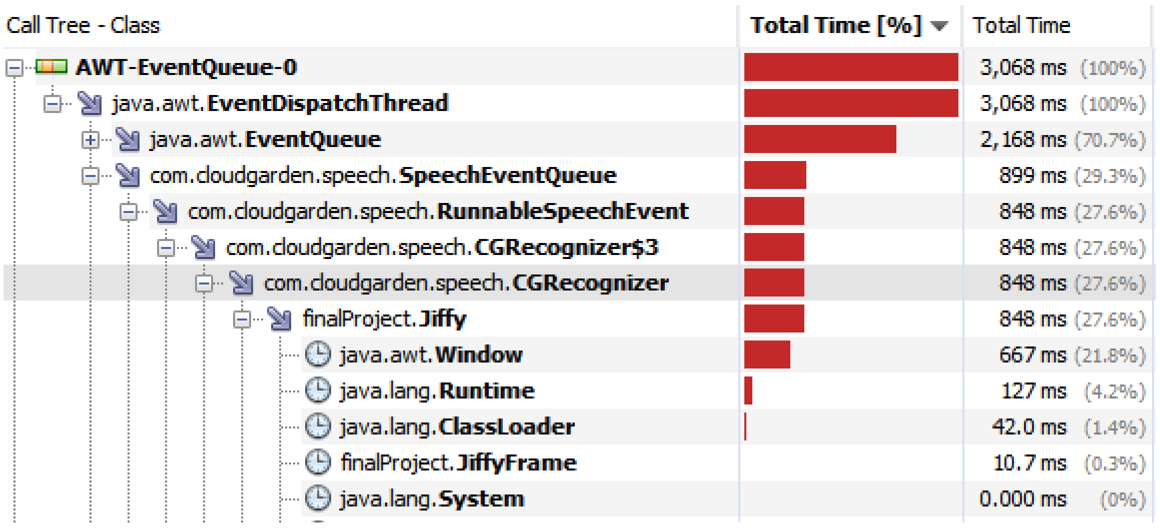
Measured response time of various classes (Focus on the highlighted part):

For speech Synthesis:



Call Tree - Class	Total Time [...	Total Time
AWT-EventQueue-0	🟥	1,053 ms (100%)
java.awt.**EventDispatchThread**	🟥	1,053 ms (100%)
java.awt.**EventQueue**	🟥	1,043 ms (99.1%)
com.cloudgarden.speech.**SpeechEventQueue**	🟥	9.77 ms (0.9%)
Self time		0.000 ms (0%)
main	🟥	151 ms (100%)
finalProject.**FinalProject**	🟥	151 ms (100%)
finalProject.**Choice**	🟥	70.5 ms (46.7%)
java.awt.**Window**	🟥	50.6 ms (33.5%)
java.awt.**Component**	🟥	20.0 ms (13.3%)
java.lang.**ClassLoader**	🟥	9.79 ms (6.5%)
Self time		0.000 ms (0%)

For speech Recognition:



Call Tree - Class	Total Time [%]	Total Time
AWT-EventQueue-0	🟥	3,068 ms (100%)
java.awt.**EventDispatchThread**	🟥	3,068 ms (100%)
java.awt.**EventQueue**	🟥	2,168 ms (70.7%)
com.cloudgarden.speech.**SpeechEventQueue**	🟥	899 ms (29.3%)
com.cloudgarden.speech.**RunnableSpeechEvent**	🟥	848 ms (27.6%)
com.cloudgarden.speech.**CGRecognizer$3**	🟥	848 ms (27.6%)
com.cloudgarden.speech.**CGRecognizer**	🟥	848 ms (27.6%)
finalProject.**Jiffy**	🟥	848 ms (27.6%)
java.awt.**Window**	🟥	667 ms (21.8%)
java.lang.**Runtime**	🟥	127 ms (4.2%)
java.lang.**ClassLoader**	🟥	42.0 ms (1.4%)
finalProject.**JiffyFrame**		10.7 ms (0.3%)
java.lang.**System**		0.000 ms (0%)

Even our recognizer took only 0.8 seconds to execute or we can say that to recognise and process our speech commands.

CHAPTER 6: CONCLUSION

Major benefits upon implementation of the project involve:

- Anyone who is tired of ordinary mouse control or key-typing might find voice control useful.
- Helpful for abled-people which have difficulties in typing and mouse control. These difficulties can be due to any physical condition.
- Aids the people who tend to multitask most of the time (eg. People can listen to their documents while doing some other work or household chores).

There is always a need to modify and add features according to the growing technology. We know there is profitability for modifying the technology to compete the market. Even though there are certain disadvantages and limitations to these systems due to lack of funds and limited scope and time available to us, they can be overcome by making certain changes and enhancements to our current system.

6.2 Future Scope

On implementing the system, we can assert with confidence that such systems can be used for a variety of applications. This system can be used for a variety of other applications such as in cars for assisting the driver for manoeuvring, picking up a call, changing songs via voice while driving and many others to name a few. Unique identification of voice is often of utmost importance in many sectors of industry. The speaker recognition system can be used for various security applications such as password authentication of bank account or homes or even high profile offices. The algorithms used in this project are uniformly accepted and recognized and are used for practical implementation of such systems. The market for voice implemented systems is continuously growing and need for systems like these are also increasing.

References

Books and Research Papers:

[1] Ksenia Shalonova, “Automatic Speech Recognition” 07 DEC 2007 Source:http://www.cs.bris.ac.uk/Teaching/Resources/COMS12303/lectures/Ksenia-Shalonova-Speech-Recognition.pdf

[3] "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.

[11] Panikos Heracleous, Hiroshi Ishiguro and Norihiro Hagita, ―Visual-speech to text conversion applicable to telephone communication for deaf individuals_ 18th International Conference on Telecommunication 2011. Pp 130-133

[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

[14] Tony Ayres and Brian Nolan of theInstitute of Technology Blanchardstown, Ireland, “Voice activated command and control with speech recognition”, ELSEVIER, August, 2005.

[19] Willie Walker, Paul Lamere and Philip Kwok, Sun Microsystems Laboratories Burlington, Massachusetts, “FreeTTS - A Performance Case Study”, August, 2002, Sun Microsystems.

Websites:

[2] http://www.abilityhub.com/speech/speech-description.htm March 7, 2017

[7] Charu Joshi “Speech Recognition” March 8, 2017
Source: http://www.scribd.com/doc/2586608/speechrecognition.pdf March 13, 2017

[8] John Kirriemuir “Speech recognition technologies” March 18, 2017

[9] http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speechrecognition.htm/printable last updated. March 7, 2017

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

[11] http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition3.ht April 15, 2017

[12] http://en.wikipedia.org/wiki/Speech_recognition April 15, 2017

[16]: Siri Problems- www.macworld.co.uk/news/iphone/siri-troubleshooting-guide-14-worst-siri-annoyances3490680. April 18, 2017

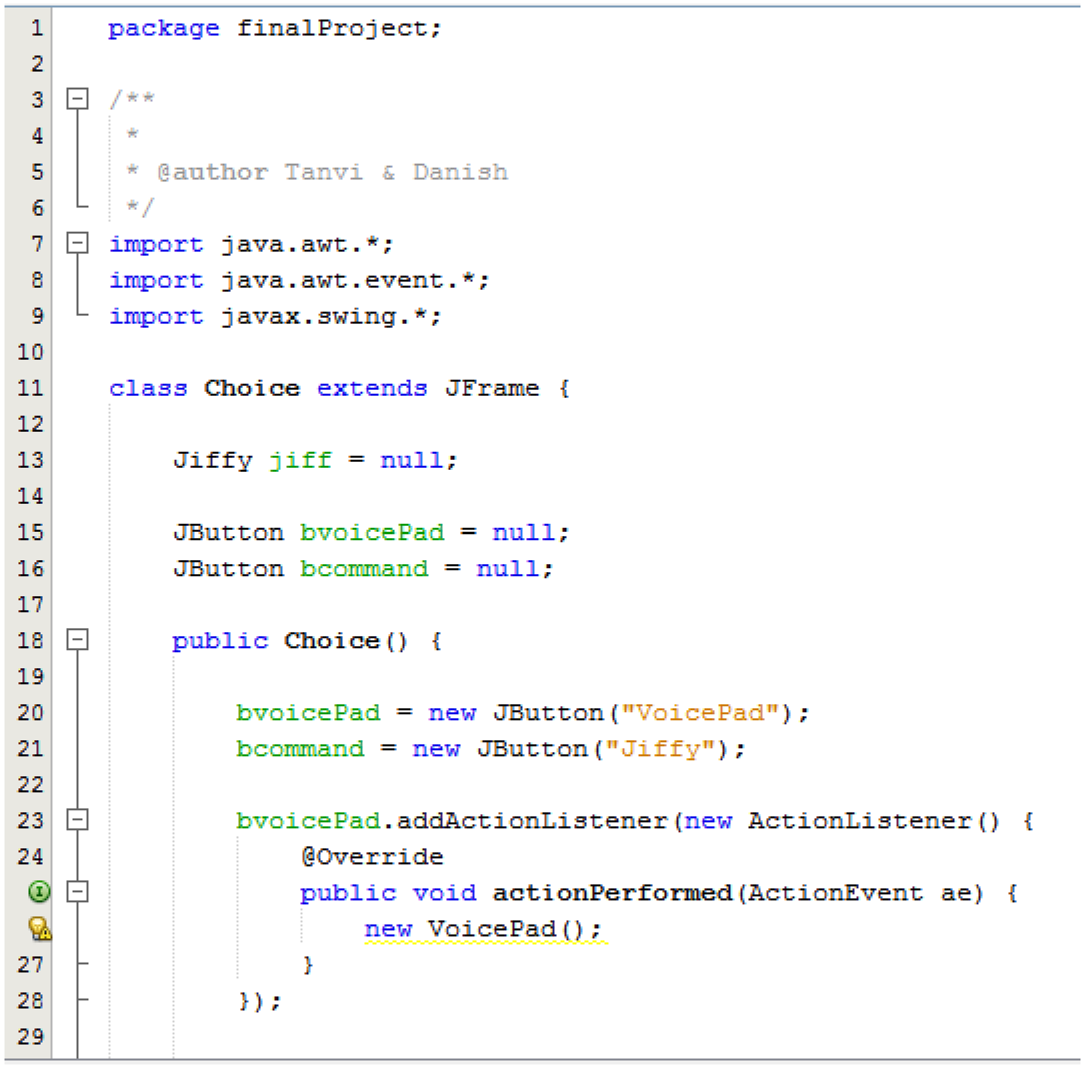
[17]: Windows Speech- forums.windowscentral.com/cortana/281601-anyone-else-having-voice-recognition-problems-cortana-today.html. April 20, 2017

[18]: Java Speech API- docs.oracle.com/cd/E17802_01/products/products/java-media/speech/forDevelopers/jsapi-doc/. April 24, 2017

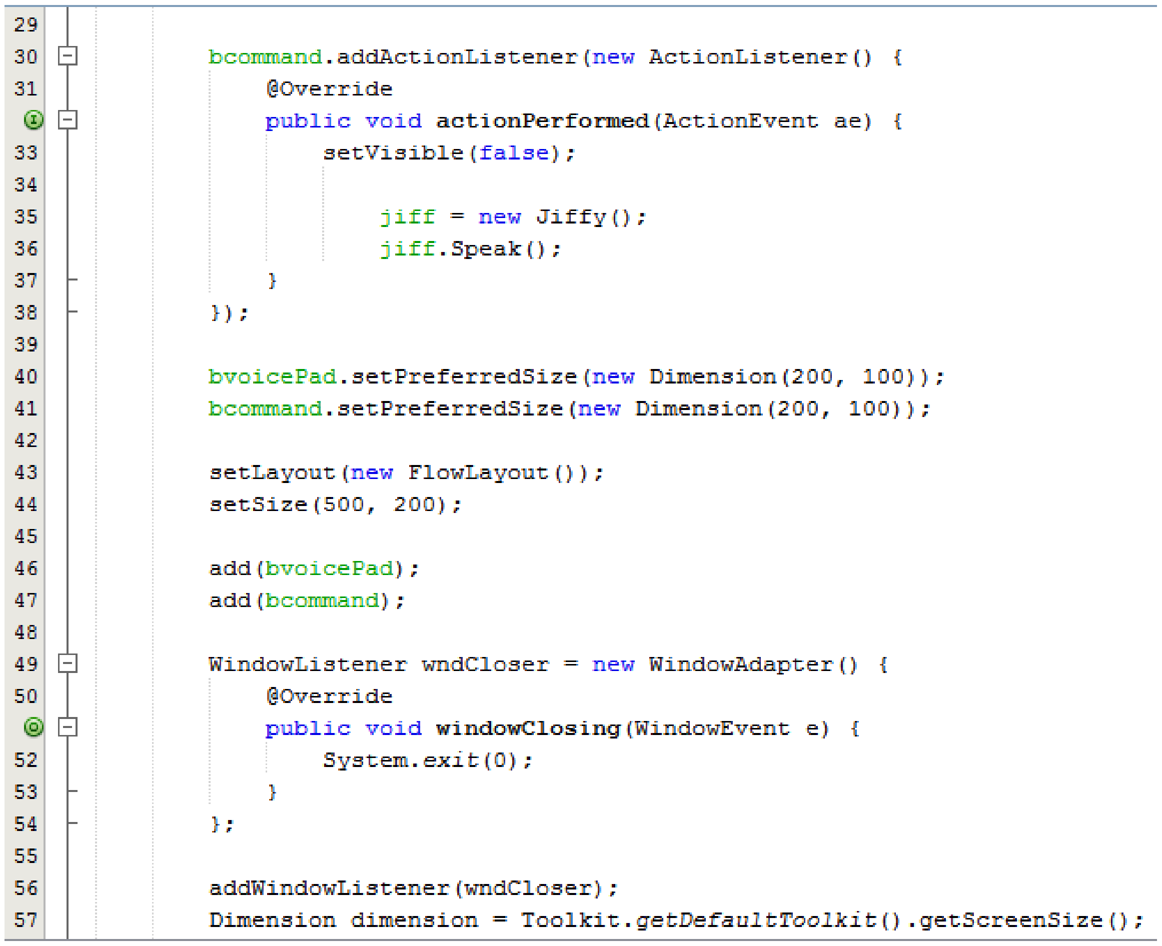
[19]: Java Speech- http://half-wit4u.blogspot.in/2011/02/speech-to-text-using-java-api.html. April 24, 2017

Appendices

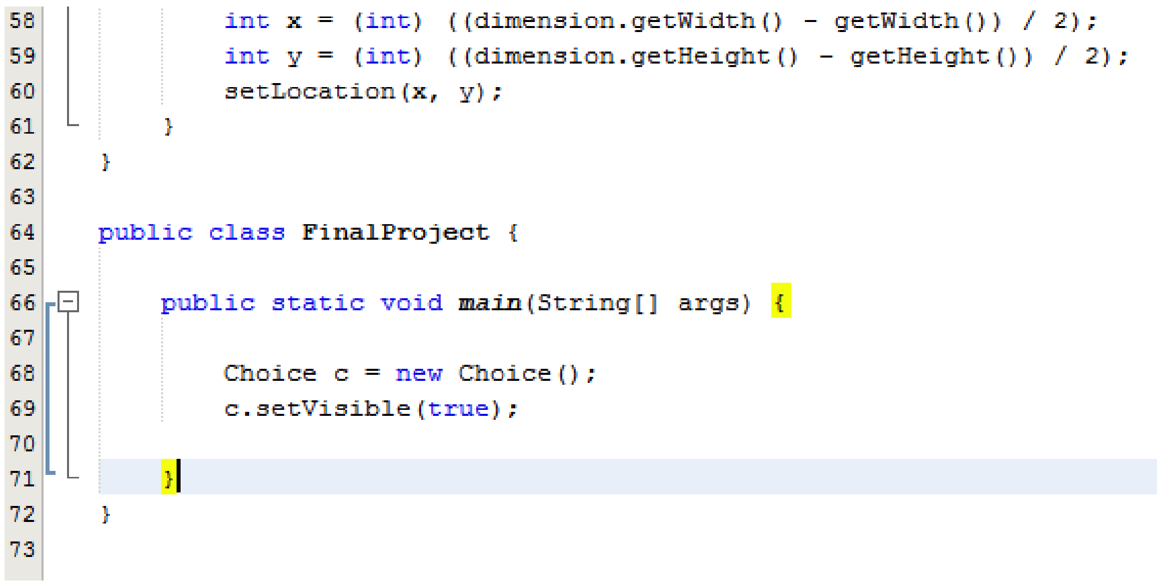
CODE:



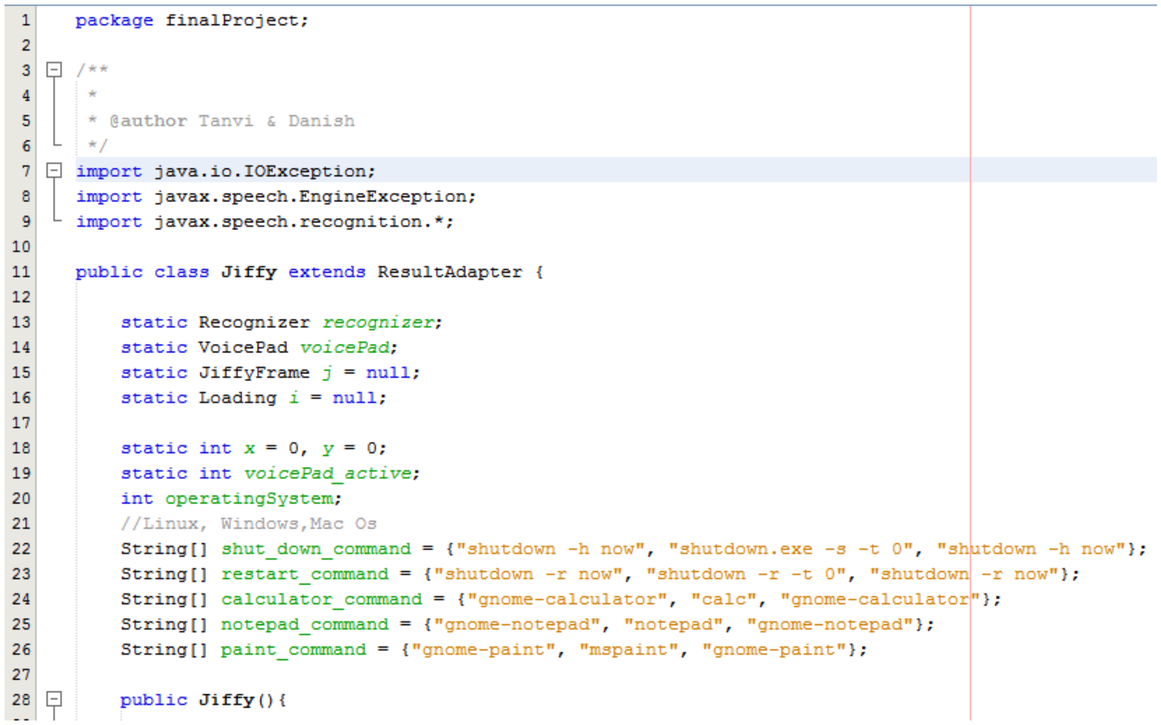
```
package finalProject;

/**
 *
 * @author Tanvi & Danish
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Choice extends JFrame {

    Jiffy jiff = null;

    JButton bvoicePad = null;
    JButton bcommand = null;

    public Choice() {

        bvoicePad = new JButton("VoicePad");
        bcommand = new JButton("Jiffy");

        bvoicePad.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
                new VoicePad();
            }
        });

```



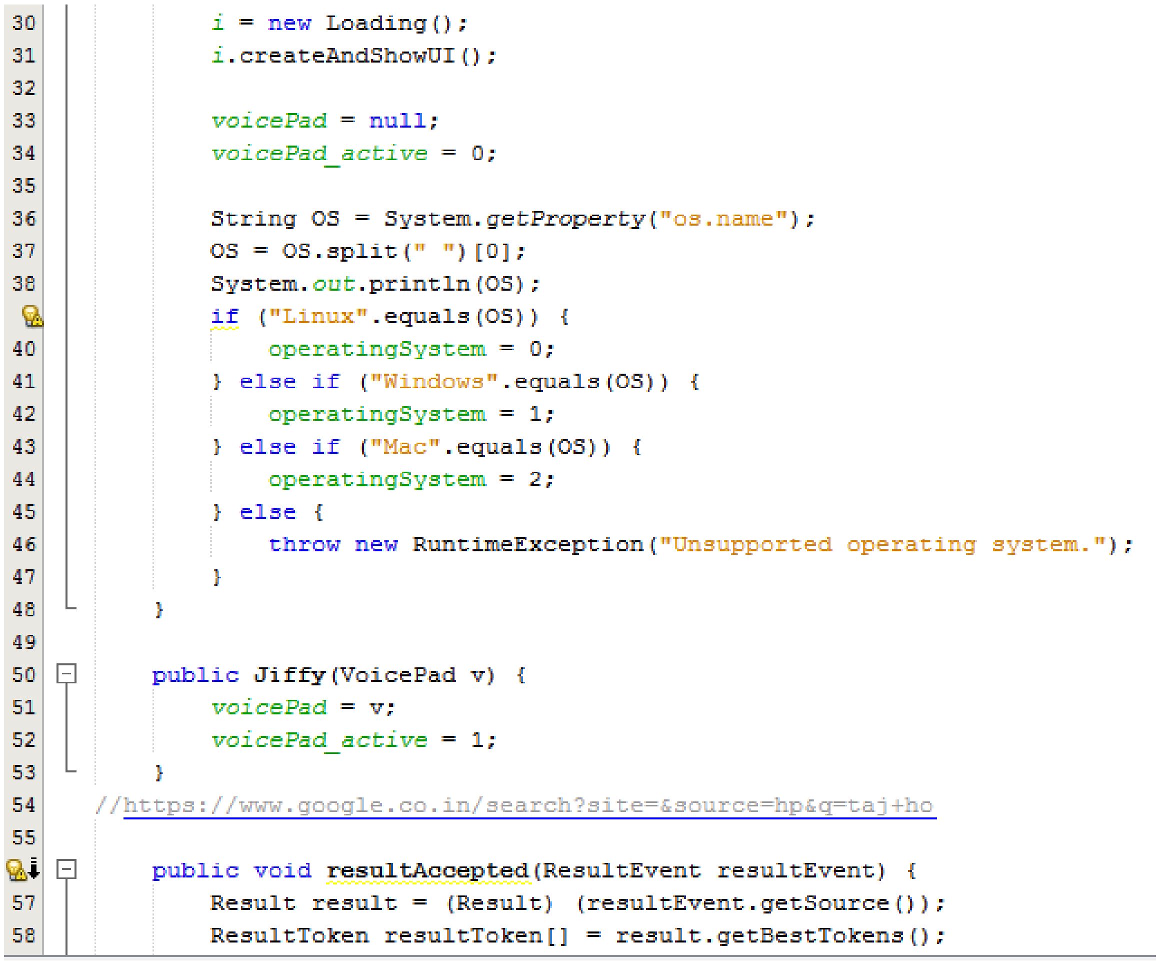
```java
        bcommand.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
                setVisible(false);

                    jiff = new Jiffy();
                    jiff.Speak();
            }
        });

        bvoicePad.setPreferredSize(new Dimension(200, 100));
        bcommand.setPreferredSize(new Dimension(200, 100));

        setLayout(new FlowLayout());
        setSize(500, 200);

        add(bvoicePad);
        add(bcommand);

        WindowListener wndCloser = new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        };

        addWindowListener(wndCloser);
        Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
```



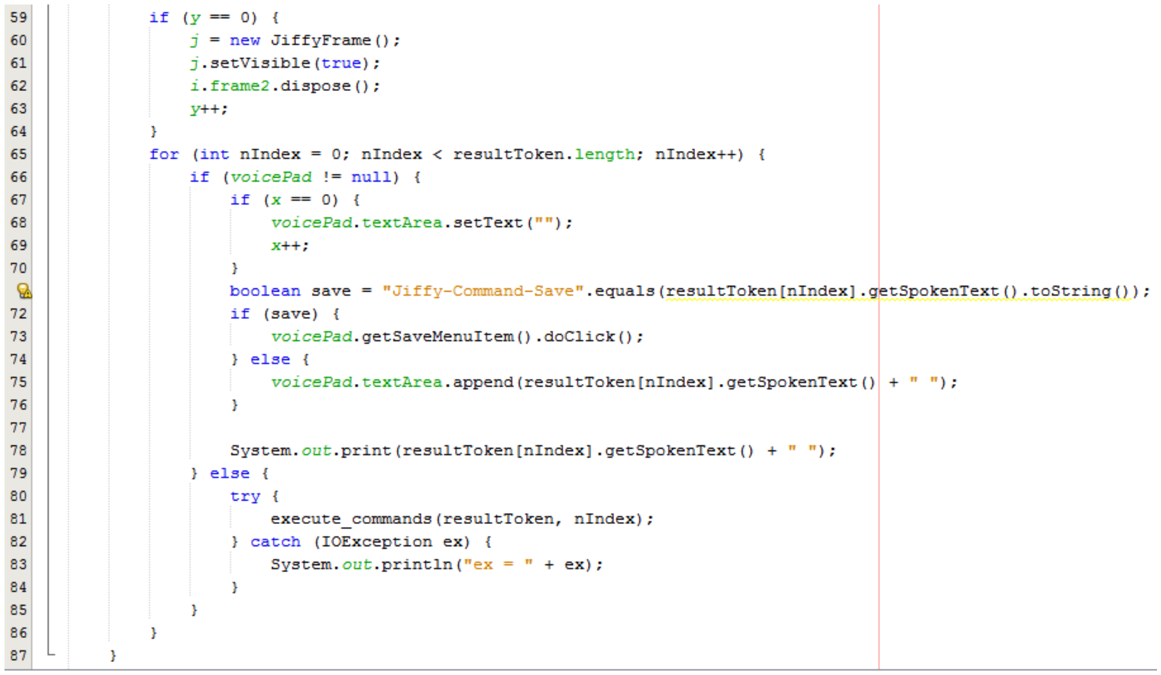
```java
        int x = (int) ((dimension.getWidth() - getWidth()) / 2);
        int y = (int) ((dimension.getHeight() - getHeight()) / 2);
        setLocation(x, y);
    }
}

public class FinalProject {

    public static void main(String[] args) {

        Choice c = new Choice();
        c.setVisible(true);

    }
}
```



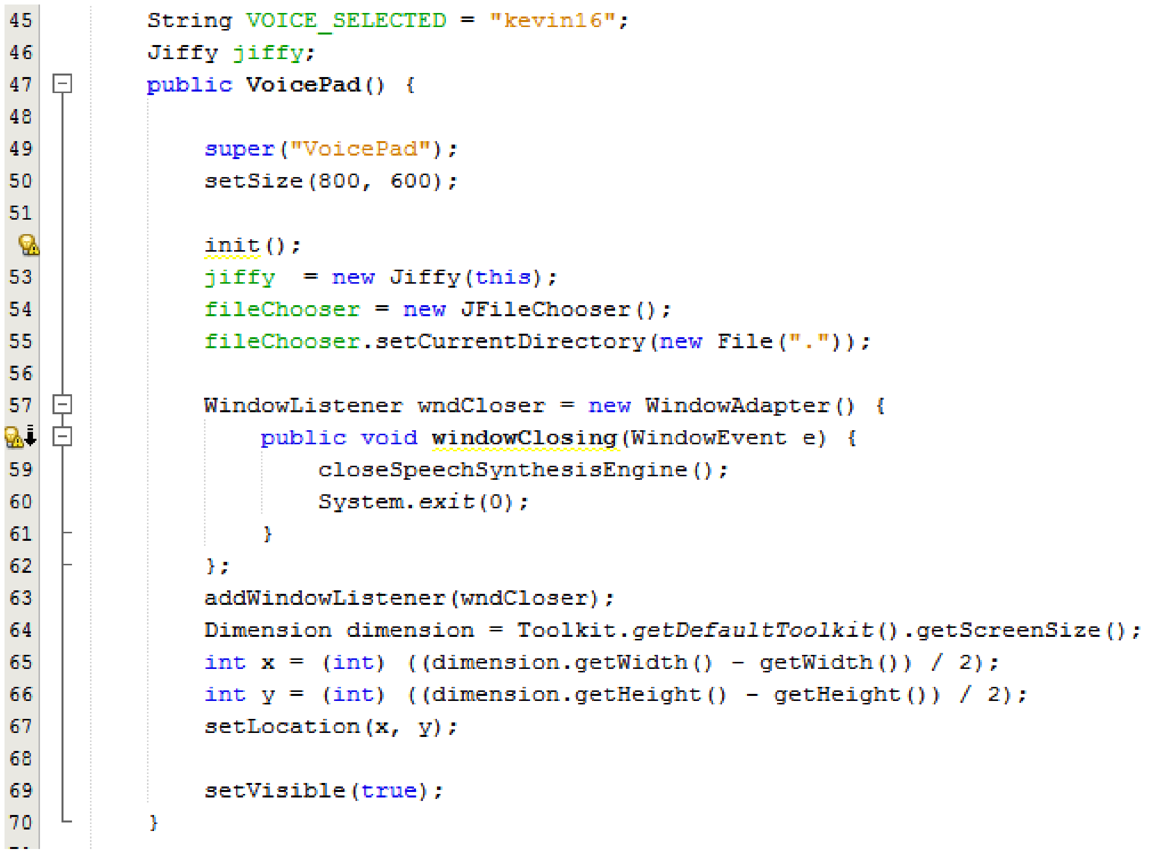
```java
package finalProject;

/**
 *
 * @author Tanvi & Danish
 */
import java.io.IOException;
import javax.speech.EngineException;
import javax.speech.recognition.*;

public class Jiffy extends ResultAdapter {

    static Recognizer recognizer;
    static VoicePad voicePad;
    static JiffyFrame j = null;
    static Loading i = null;

    static int x = 0, y = 0;
    static int voicePad_active;
    int operatingSystem;
    //Linux, Windows,Mac Os
    String[] shut_down_command = {"shutdown -h now", "shutdown.exe -s -t 0", "shutdown -h now"};
    String[] restart_command = {"shutdown -r now", "shutdown -r -t 0", "shutdown -r now"};
    String[] calculator_command = {"gnome-calculator", "calc", "gnome-calculator"};
    String[] notepad_command = {"gnome-notepad", "notepad", "gnome-notepad"};
    String[] paint_command = {"gnome-paint", "mspaint", "gnome-paint"};

    public Jiffy(){
```



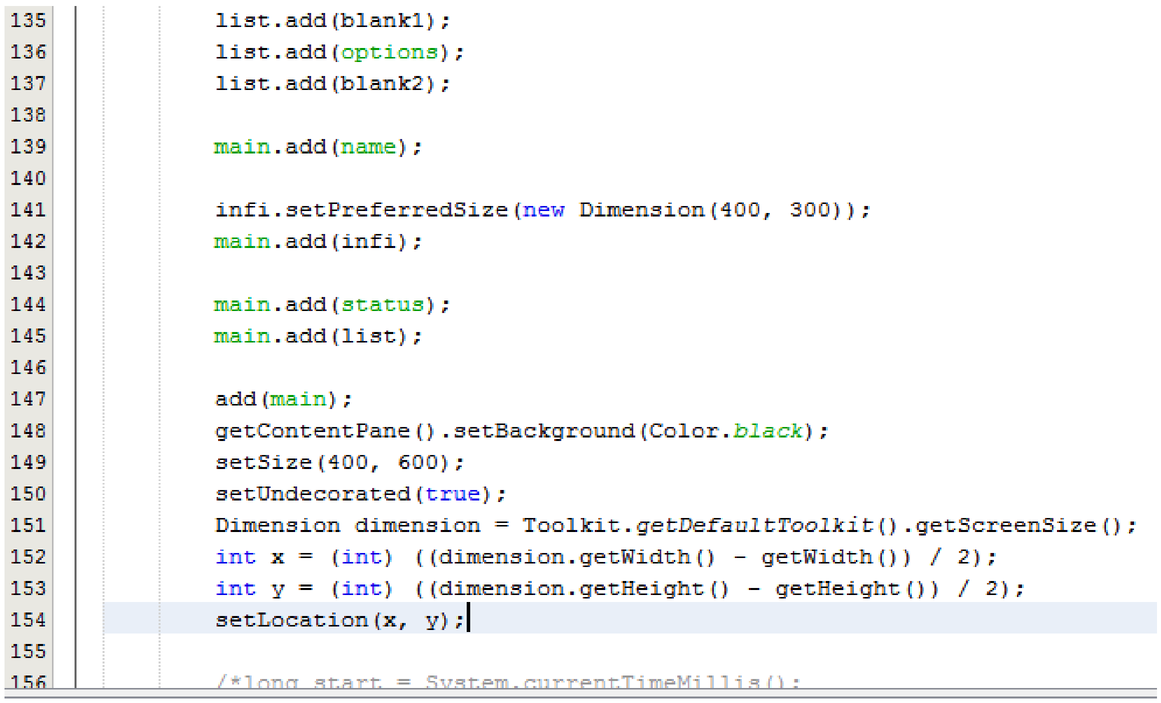
```
        i = new Loading();
        i.createAndShowUI();

        voicePad = null;
        voicePad_active = 0;

        String OS = System.getProperty("os.name");
        OS = OS.split(" ")[0];
        System.out.println(OS);
        if ("Linux".equals(OS)) {
            operatingSystem = 0;
        } else if ("Windows".equals(OS)) {
            operatingSystem = 1;
        } else if ("Mac".equals(OS)) {
            operatingSystem = 2;
        } else {
            throw new RuntimeException("Unsupported operating system.");
        }
    }

    public Jiffy(VoicePad v) {
        voicePad = v;
        voicePad_active = 1;
    }
//https://www.google.co.in/search?site=&source=hp&q=taj+ho

    public void resultAccepted(ResultEvent resultEvent) {
        Result result = (Result) (resultEvent.getSource());
        ResultToken resultToken[] = result.getBestTokens();
```



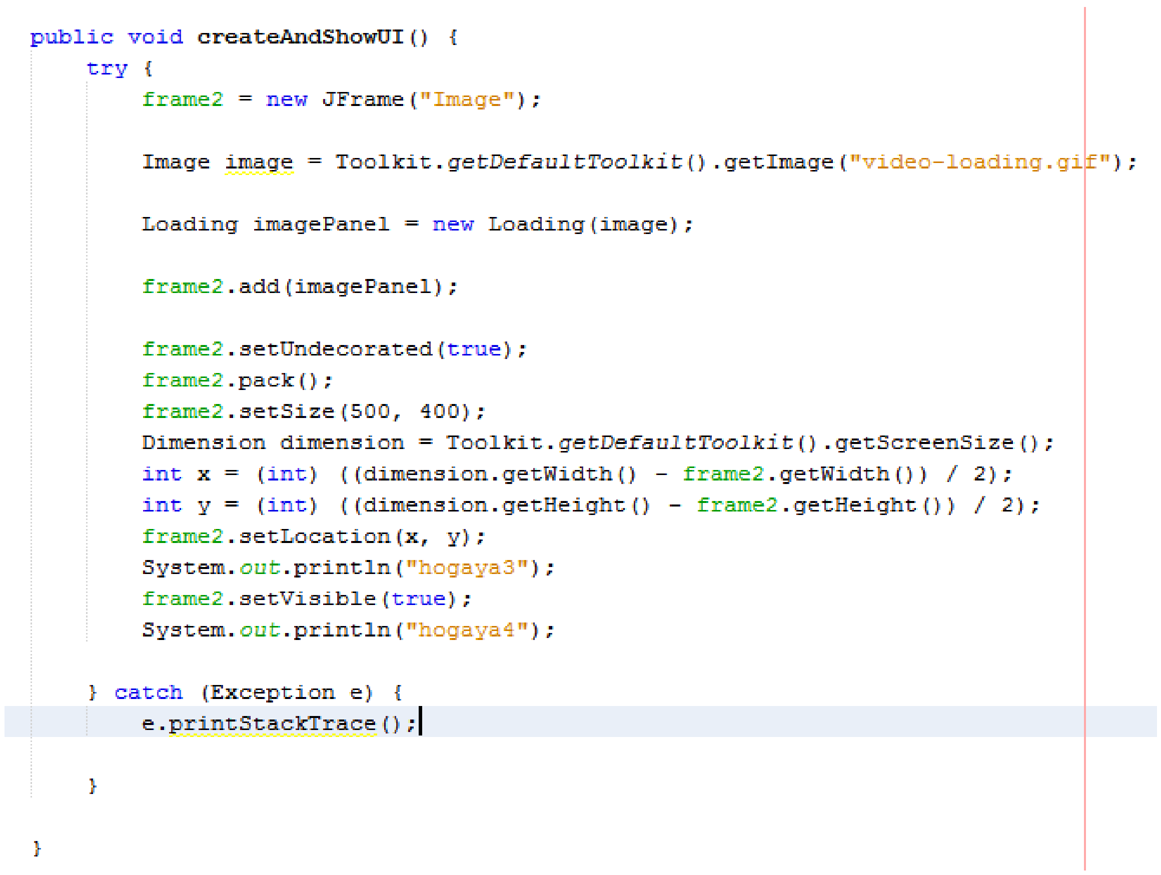
```java
            if (y == 0) {
                j = new JiffyFrame();
                j.setVisible(true);
                i.frame2.dispose();
                y++;
            }
            for (int nIndex = 0; nIndex < resultToken.length; nIndex++) {
                if (voicePad != null) {
                    if (x == 0) {
                        voicePad.textArea.setText("");
                        x++;
                    }
                    boolean save = "Jiffy-Command-Save".equals(resultToken[nIndex].getSpokenText().toString());
                    if (save) {
                        voicePad.getSaveMenuItem().doClick();
                    } else {
                        voicePad.textArea.append(resultToken[nIndex].getSpokenText() + " ");
                    }

                    System.out.print(resultToken[nIndex].getSpokenText() + " ");
                } else {
                    try {
                        execute_commands(resultToken, nIndex);
                    } catch (IOException ex) {
                        System.out.println("ex = " + ex);
                    }
                }
            }
        }
```



```java
    String VOICE_SELECTED = "kevin16";
    Jiffy jiffy;
    public VoicePad() {

        super("VoicePad");
        setSize(800, 600);

        init();
        jiffy  = new Jiffy(this);
        fileChooser = new JFileChooser();
        fileChooser.setCurrentDirectory(new File("."));

        WindowListener wndCloser = new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                closeSpeechSynthesisEngine();
                System.exit(0);
            }
        };
        addWindowListener(wndCloser);
        Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (int) ((dimension.getWidth() - getWidth()) / 2);
        int y = (int) ((dimension.getHeight() - getHeight()) / 2);
        setLocation(x, y);

        setVisible(true);
    }
```



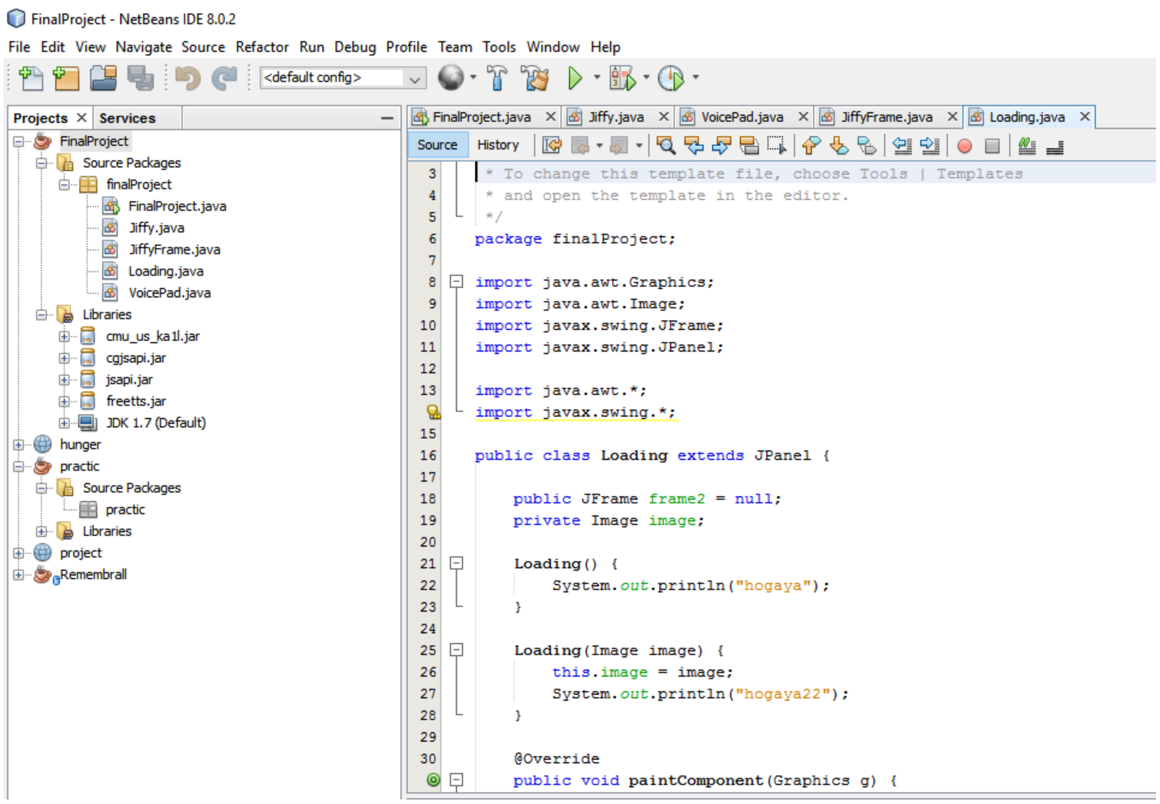
```
        list.add(blank1);
        list.add(options);
        list.add(blank2);

        main.add(name);

        infi.setPreferredSize(new Dimension(400, 300));
        main.add(infi);

        main.add(status);
        main.add(list);

        add(main);
        getContentPane().setBackground(Color.black);
        setSize(400, 600);
        setUndecorated(true);
        Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (int) ((dimension.getWidth() - getWidth()) / 2);
        int y = (int) ((dimension.getHeight() - getHeight()) / 2);
        setLocation(x, y);

        /*long start = System.currentTimeMillis();
```



```java
public void createAndShowUI() {
    try {
        frame2 = new JFrame("Image");

        Image image = Toolkit.getDefaultToolkit().getImage("video-loading.gif");

        Loading imagePanel = new Loading(image);

        frame2.add(imagePanel);

        frame2.setUndecorated(true);
        frame2.pack();
        frame2.setSize(500, 400);
        Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (int) ((dimension.getWidth() - frame2.getWidth()) / 2);
        int y = (int) ((dimension.getHeight() - frame2.getHeight()) / 2);
        frame2.setLocation(x, y);
        System.out.println("hogaya3");
        frame2.setVisible(true);
        System.out.println("hogaya4");

    } catch (Exception e) {
        e.printStackTrace();

    }

}
```

Screenshots:

Figure A.1: Use of NetBeans:



```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalProject;

import java.awt.Graphics;
import java.awt.Image;
import javax.swing.JFrame;
import javax.swing.JPanel;

import java.awt.*;
import javax.swing.*;

public class Loading extends JPanel {

    public JFrame frame2 = null;
    private Image image;

    Loading() {
        System.out.println("hogaya");
    }

    Loading(Image image) {
        this.image = image;
        System.out.println("hogaya22");
    }

    @Override
    public void paintComponent(Graphics g) {
```

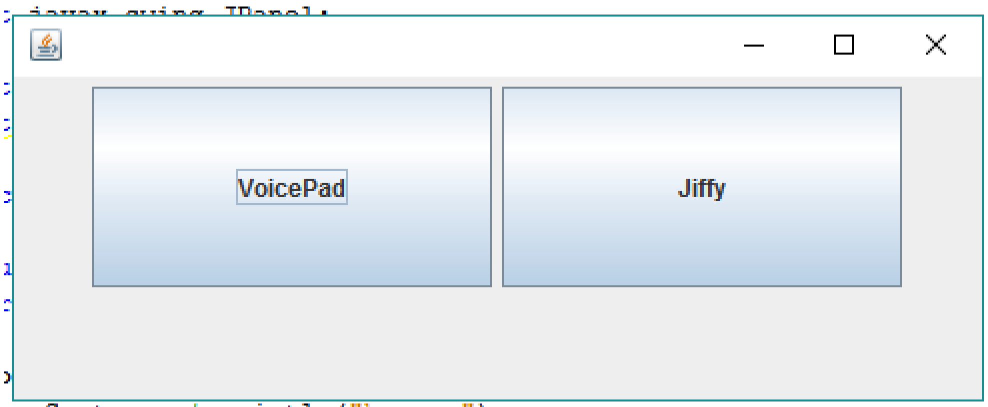
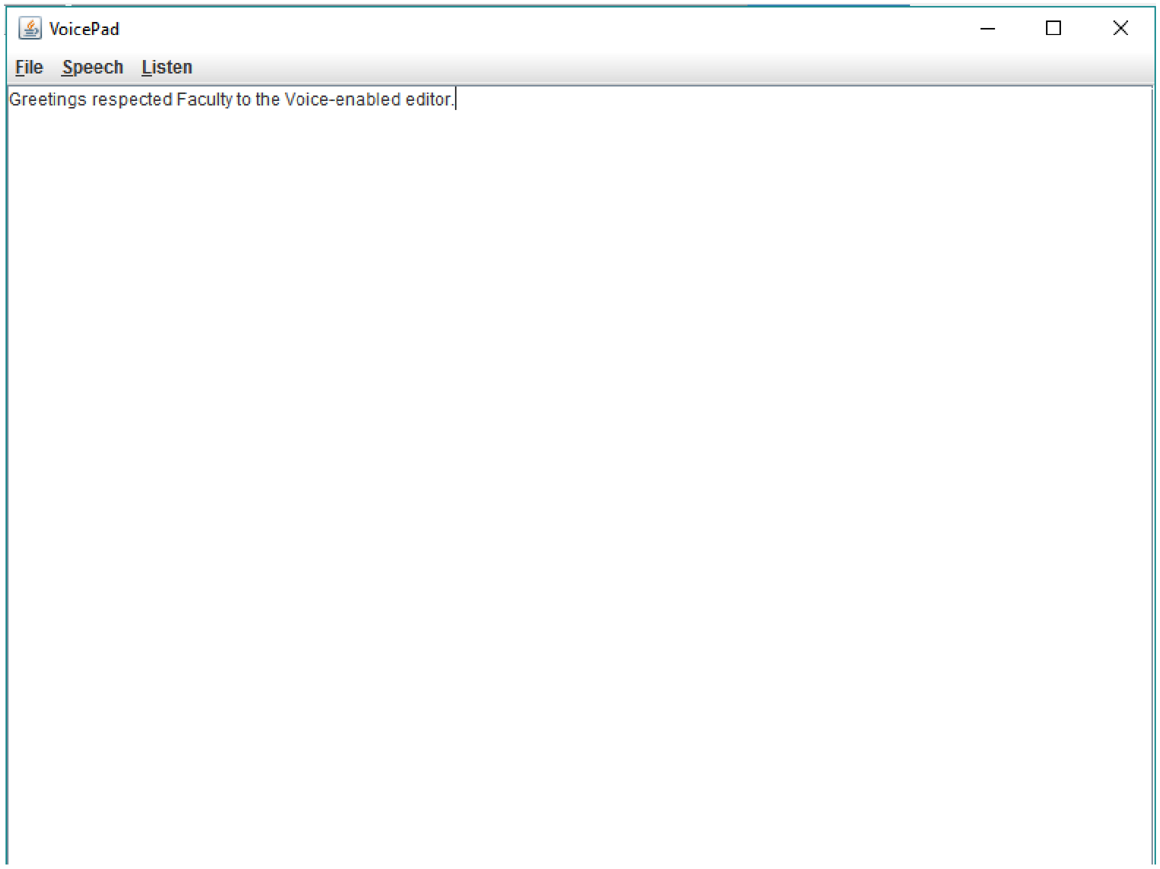
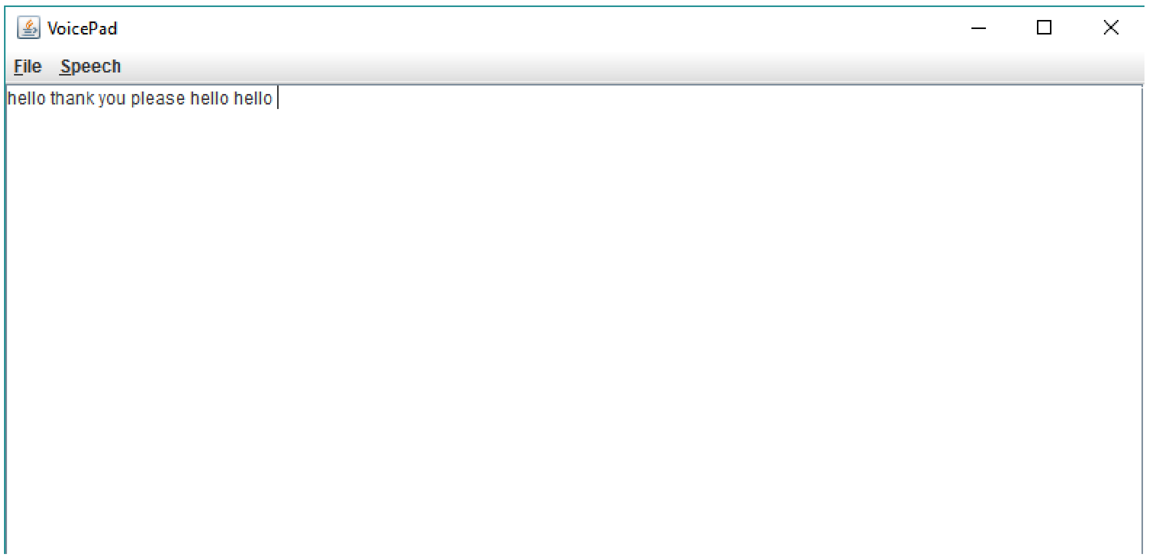


Figure A.2: VoicePad



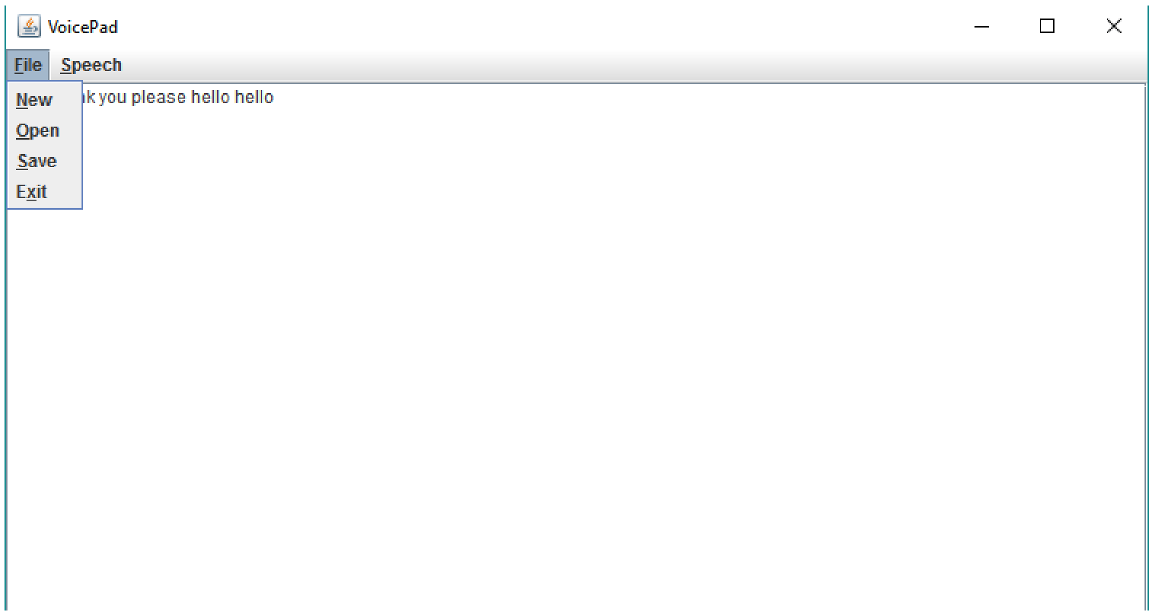
Greetings respected Faculty to the Voice-enabled editor.

Figure A.3: VoicePad in action



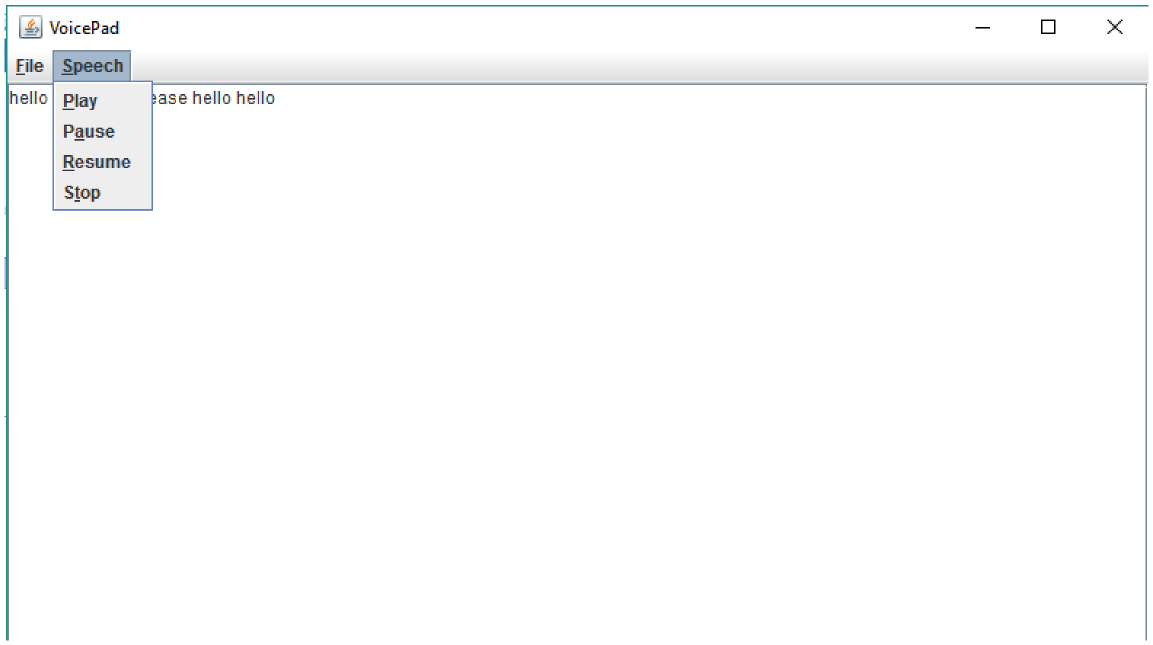
hello thank you please hello hello

Figure A.4: File Menu



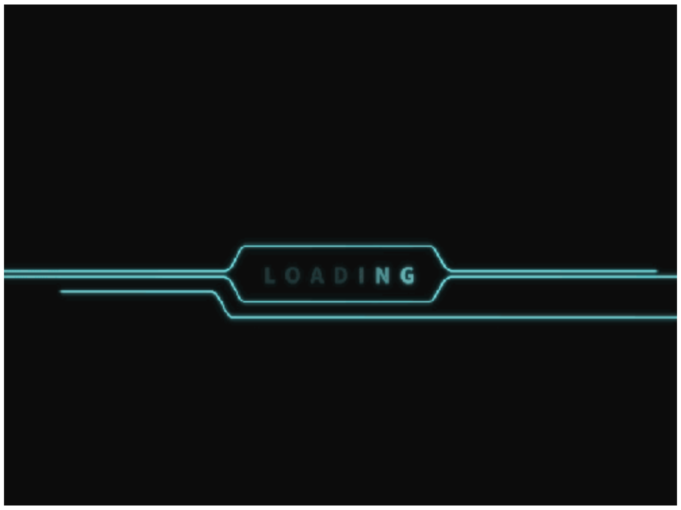
ık you please hello hello

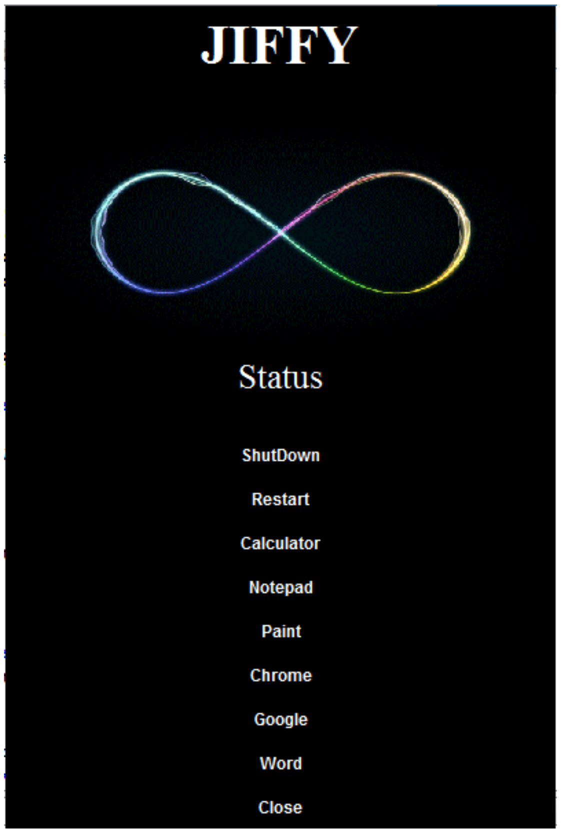
Figure A.5: Speech Menu



hello ... ease hello hello

Figure A.6: Jiffy




JIFFY
Status
ShutDown
Restart
Calculator
Notepad
Paint
Chrome
Google
Word
Close