SPEECH RECOGNITION SYSTEM

Minor-project

Submitted in the partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

COMPUTER SCIENCE

By

DEEPAK(15013022**)**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GURU JAMBHESHWAR UNIVERSITY OF SCIENCE & TECHNOLOGY, HISAR

ACKNOWLEDGEMENT

This project was undertaken at **GJUST,Hisar** during Summer Internship engineering to automate the system .The project is named as **Voice Recognition System.**

The purpose of this report is to assemble under one cover a sufficient body of knowledge about management and development a successful software engineering project. The following quotes outline the basic idea behind this technical report.

This report is about the adaptation of the techniques of project development and reflects the practice and methods of software engineering project this report is intended for:

• *Project managers*—the report delivers the necessary information of the process a software development project

• *Project coordinators*—the tutorial presents the state of the practice in software development and management techniques.

• *Software engineers, programmers, analysts, and other computer personnel* —the report contains a general description of—and problems in—software engineering project development, plus a number of methodologies and techniques for managing a software development project.

Project Developed By:

DEEPAK

B.Tech CSE(B-1st)

Roll :-15013022

Content's Table

INTODUCTION

Speech recognition is the inter-disciplinary sub-field of computational linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers. It is also known as "automatic speech recognition" (ASR), "computer speech recognition", or just "speech to text" (STT). It incorporates knowledge and research in the linguistics, computer science, and electrical engineering fields.

Speech recognition applications include voice user interfaces such as voice dialing (e.g. "Call home"), call routing (e.g. "I would like to make a collect call"), domotic appliance control, search (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed direct voice input).

In daily life people make voice recognition and speech recognition same things while voice recognition refers to identifying the speaker, rather than what they are saying refers to speech recognition.

REVIEW TO TECHNOLOGY

CORE JAVA : Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.

Characteristics:

- Simple
- Object oriented
- Distributed
- Interpreted
- Robust
- Secure
- Architecture neutral
- Portable
- Dynamic
- Threaded

What Can Java Technology Do…?

The most common types of programs written in the Java programming language are applets and applications. If you've accessed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

SWING : Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

Swing is a platform-independent, "model-view-controller" GUI framework for Java, which follows a single-threaded programming model.[5] Additionally, this framework provides a layer of abstraction between the code structure and graphic presentation of a Swing-based GUI.

PROPERTIES :

- Extensible
- Configurable
- LightWeight UI
- Loosely Coupled
- Model View Controller
- Accessibility API
- Java 2D™ API (Java 2 Platform only)
- Drag and Drop Support (Java 2 Platform only)

JAVA SPEECH API : The **Java Speech API** (JSAPI) is an application programming interface for cross-platform support of command and control recognizers, dictation systems, and speech synthesizers. Although JSAPI defines an interface only there are several implementations created by third parties, for example FreeTTS.

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

- Structure analysis: Processes 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: Analyzes the input text for special constructs of the language. In English, special treatment is required for abbreviations, acronyms, dates, times, numbers, currency amounts, e-mail 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: Converts each word to phonemes. A phoneme is a basic unit of sound in a language.
- Prosody analysis: Processes the sentence structure, words, and phonemes to determine the appropriate prosody for the sentence.
- Waveform production: Uses the phonemes and prosody information to produce the audio waveform for each sentence.

Speech synthesizers can make errors in any of the processing steps described above. Human ears are well-tuned to detecting these errors, but careful work by developers can minimize errors and improve the speech output quality. While the Java Speech API 1 relied on the Java Speech API Markup Language (JSML), the newer release utilizes SSML to provide many ways for you to improve the output quality of a speech synthesizer.

The major steps of a typical speech recognizer are as follows:

- Grammar design: Defines the words that may be spoken by a user and the patterns in which they may be spoken.
- Signal processing: Analyzes the spectrum (i.e., the frequency) characteristics of the incoming audio.
- Phoneme recognition: Compares the spectrum patterns to the patterns of the phonemes of the language being recognized.
- Word recognition: Compares the sequence of likely phonemes against the words and patterns of words specified by the active grammars.
- Result generation: Provides the application with information about the words the recognizer has detected in the incoming audio.

A *grammar* is an object in the Java Speech API that indicates what words a user is expected to say and in what patterns those words may occur. Grammars are important to speech recognizers because they constrain the recognition process. These constraints make recognition faster and more accurate because the recognizer does not have to check for bizarre sentences.

REQUIREMENTS

SOFTWARE REQUIREMENTS :

- Java Development Kit (JDK 1.8)
- Operating System: Any operating system

HARDWARE REQUIREMENTS :

- Processor: Intel Pentium 4 or more
- RAM: 64 MB or more
- Hard Disk: 4GB or more.

SOURCE CODE

SpeechToTextConverter :

```
package com.sarf.talkingjava;

import javax.speech.Central;

import javax.speech.recognition.*;

import java.io.FileReader;

import java.util.Locale;


class SpeechToTextConverter extends ResultAdapter {

VoiceReco r;

 static Recognizer recognizer;


    public void resultAccepted(ResultEvent resultEvent) {

        Result result = (Result)(resultEvent.getSource());

        ResultToken resultToken[] = result.getBestTokens();


        for (int nIndex = 0; nIndex < resultToken.length; nIndex++){

            System.out.println(resultToken[nIndex].getSpokenText() + " ");

        }

        try {


            r.jta.setText("Run Successfully");
```

```
            // Deallocate the recognizer
            recognizer.forceFinalize(true);
            recognizer.deallocate();
        }catch (Exception exception) {
            exception.printStackTrace();
        }
        System.exit(0);
    }

    SpeechToTextConverter() {
        try {
                    this.r=r;
            Central.registerEngineCentral
               ("com.cloudgarden.speech.CGEngineCentral");
            RecognizerModeDesc desc =
               new RecognizerModeDesc(Locale.US,Boolean.TRUE);
            // Create a recognizer that supports US English.
            recognizer = Central.createRecognizer(desc);

            // Start up the recognizer
            recognizer.allocate();

            // Load the grammar from a file, and enable it
            FileReader fileReader =
              new FileReader("E:\\com\\sarf\\talkingjava\\example\\helloWorld.gram");
```

```
        RuleGrammar grammar = recognizer.loadJSGF(fileReader);
        grammar.setEnabled(true);

        // Add the listener to get results
        recognizer.addResultListener(this);

        // Commit the grammar
        recognizer.commitChanges();
        recognizer.waitEngineState(Recognizer.LISTENING);

        // Request focus and start listening
        recognizer.requestFocus();
        recognizer.resume();

        recognizer.waitEngineState(Recognizer.FOCUS_ON);

        recognizer.forceFinalize(true);
        recognizer.waitEngineState(Recognizer.DEALLOCATED);


    } catch (Exception e) {
        e.printStackTrace();
        System.exit;}}
```

VoiceReco :

```
package com.sarf.talkingjava;

import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.io.*;

import java.util.*;


public class VoiceReco implements ActionListener

{

   JFrame jf;

   JButton jb;

   JTextArea jta;

   JScrollPane jsp;



            VoiceReco()

          {

          jf = new JFrame(" Voice Recognization");

          jf.setLayout(null);


          jb = new JButton("Mic");

          jb.setBounds(200,430,80,25);

jb.addActionListener(this);
```

```
        jta = new JTextArea(50,50);


        jta.setFont(new Font("varinda",Font.PLAIN,15));


        jsp = new JScrollPane(jta);
        jsp.setBounds(10,60,500,300);


        jf.add(jsp);
        jf.add(jb);




        jf.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        jf.setSize(550,550);
        jf.setVisible(true);
        }
public void actionPerformed(ActionEvent e)
{
        if(e.getSource()==jb)
        {
        new SpeechToTextConverter();
        }
```

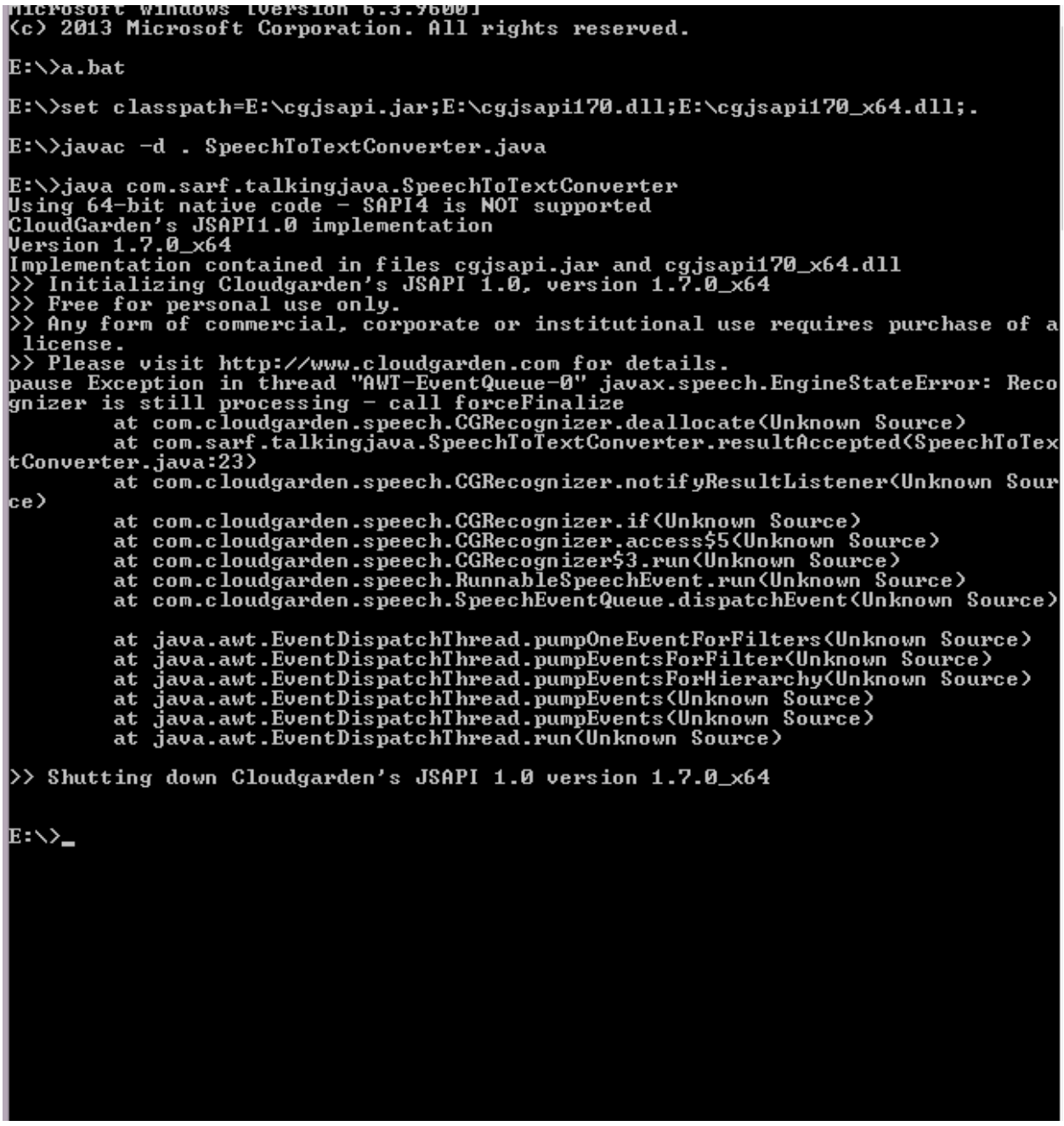
.Nbattrs :

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

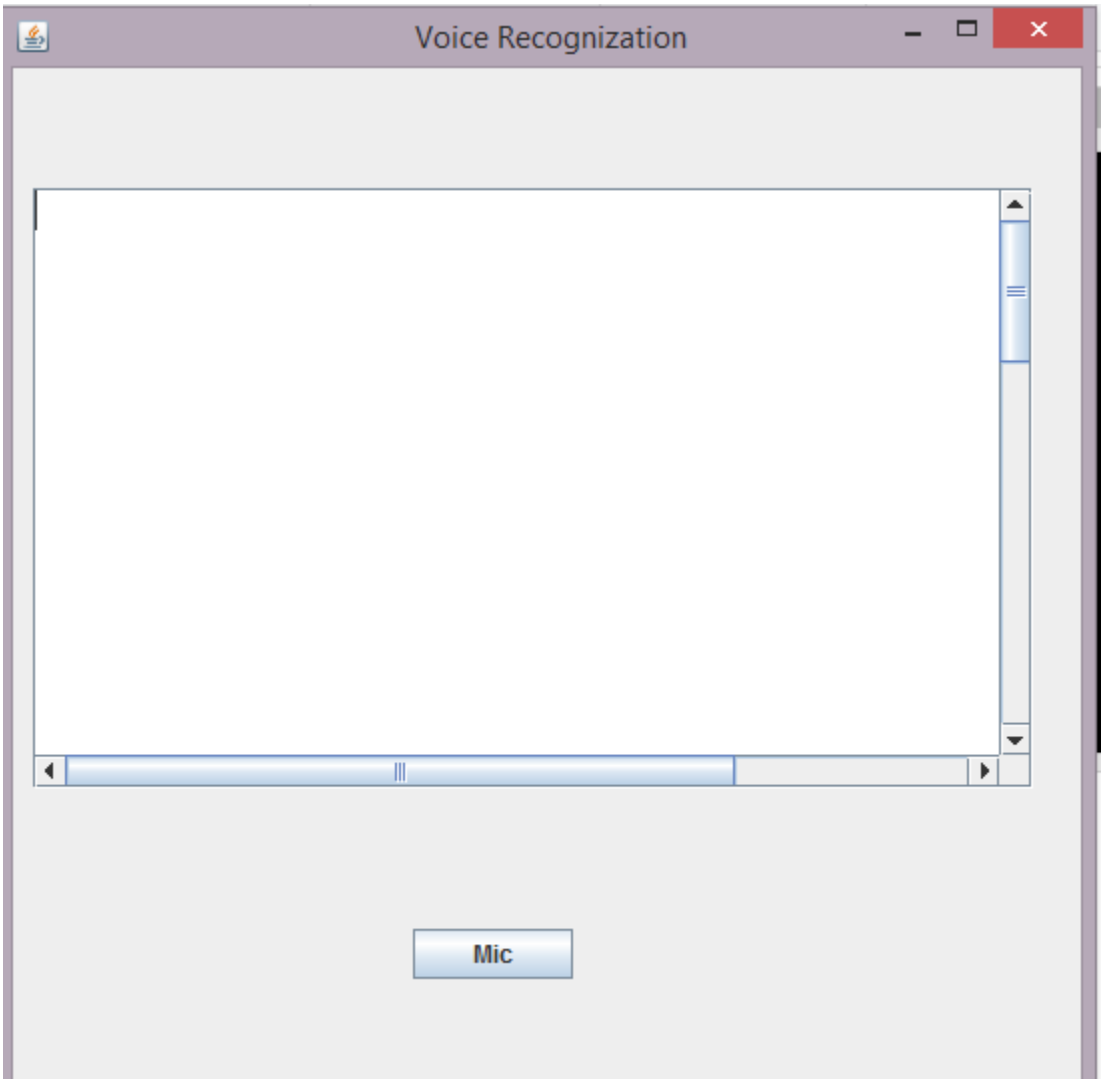
```
<!DOCTYPE attributes PUBLIC "-//NetBeans//DTD DefaultAttributes 1.0//EN"
"http://www.netbeans.org/dtds/attributes-1_0.dtd">

<attributes version="1.0">

  <fileobject name="testGram.gram">

    <attr name="org.netbeans.modules.text.IsTextFile" boolvalue="true"/>

  </fileobject>

  <fileobject name="helloWorld_test.gram">

    <attr name="org.netbeans.modules.text.IsTextFile" boolvalue="true"/>

  </fileobject>

  <fileobject name="numbers.gram">

    <attr name="org.netbeans.modules.text.IsTextFile" boolvalue="true"/>

  </fileobject>

  <fileobject name="error_test.gram">

    <attr name="org.netbeans.modules.text.IsTextFile" boolvalue="true"/>

  </fileobject>

  <fileobject name="null_void_test.gram">

    <attr name="org.netbeans.modules.text.IsTextFile" boolvalue="true"/>

  </fileobject>

  <fileobject name="helloWorld.gram">

    <attr name="org.netbeans.modules.text.IsTextFile" boolvalue="true"/>

  </fileobject>

</attributes>
```

OUTPUTS

1.


Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
E:\>a.bat
E:\>set classpath=E:\cgjsapi.jar;E:\cgjsapi170.dll;E:\cgjsapi170_x64.dll;.
E:\>javac -d . SpeechToTextConverter.java
E:\>java com.sarf.talkingjava.SpeechToTextConverter
Using 64-bit native code - SAPI4 is NOT supported
CloudGarden's JSAPI1.0 implementation
Version 1.7.0_x64
Implementation contained in files cgjsapi.jar and cgjsapi170_x64.dll
>> Initializing Cloudgarden's JSAPI 1.0, version 1.7.0_x64
>> Free for personal use only.
>> Any form of commercial, corporate or institutional use requires purchase of a license.
>> Please visit http://www.cloudgarden.com for details.
pause Exception in thread "AWT-EventQueue-0" javax.speech.EngineStateError: Recognizer is still processing - call forceFinalize
at com.cloudgarden.speech.CGRecognizer.deallocate(Unknown Source)
at com.sarf.talkingjava.SpeechToTextConverter.resultAccepted(SpeechToTextConverter.java:23)
at com.cloudgarden.speech.CGRecognizer.notifyResultListener(Unknown Source)
at com.cloudgarden.speech.CGRecognizer.if(Unknown Source)
at com.cloudgarden.speech.CGRecognizer.access$5(Unknown Source)
at com.cloudgarden.speech.CGRecognizer$3.run(Unknown Source)
at com.cloudgarden.speech.RunnableSpeechEvent.run(Unknown Source)
at com.cloudgarden.speech.SpeechEventQueue.dispatchEvent(Unknown Source)
at java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown Source)
at java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
at java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown Source)
at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
at java.awt.EventDispatchThread.run(Unknown Source)
>> Shutting down Cloudgarden's JSAPI 1.0 version 1.7.0_x64
E:\>

2.



Conclusion

Thus we successfully created Speech Recognition System.The system provides a user friendly environment for the users to maintain their experience for searching via speech.

References

- www.w3schools.com
- www.wikipedia.com
- www.stackoverflow.com
- www.talkingjava.com