Voice Command based Android Java Code Generator

Submitted by:

MD. Shakerul Islam (14101018)

Hosne Mobarak (14101002)

MD. Mominul Islam (14101174)

Department of Computer Science and Engineering

Supervisor:

Suraiya Tairin, Lecturer, Department of Computer Science and Engineering (CSE)

Co-Supervisor:

Md.Saiful Islam, Lecturer, Department of Computer Science and Engineering (CSE)

Declaration

We would love to declare that this thesis is based on results we have found by ourselves. Materials and conclusions from researches that are conducted by world-wide by other researchers are mentioned in references section.

Sign of Supervisor	Sign of Co-Supervisor

Sign of Author	Sign of Author	Sign of Author

Acknowledgement

This topic was chosen by us and encouraged by our supervisor Suraiya Tairin and co-supervisor MD. Saiful Islam to carry on for our thesis. We are pretty grateful to Almighty Allah (SWT) for giving us the opportunity, knowledge and patience to complete our thesis project.

We are eternally grateful to our faculties, friends and family who helped us in every step of the way to complete this research. We would like to express our sincerest gratitude to our supervisor Suraiya Tairin and Co-supervisor MD. Saiful Islam for their continuous support, encouragement and guidance in every step of our research. Without them this research would not have been completed.

Apart from the closely connected people, we got immense amount of help from our friend Monwar Jahan Mufad regarding the programing part of this thesis project.

Abstract

The world is moving toward voiced based interaction with devices. A voice based code generator can make the work of a developer much easier and faster as well as bring hope for physically challenged programmers. Moreover, children can learn programming from very early age with the help of voice based code generator. So, we propose a new approach on mobile based application "Vocal - A voice based Code Generator". This voice based code generator is going to be an Android app so that people can use it whenever they want. Our "Vocal" will take voice based instructions from the user and then detect the intention of the user. Depending on the commands, it will create variable, loops, print statement, methods, condition etc. There will be some basic instructions for the user to make the proper command.

Keywords

Natural Language, Voice, Command, Instruction, Speak, Translate, Compile

Table of Contents

Table of Figures

Chapter 1
Introduction

This section will give overall short idea of our thesis project and work. Apart from our work we have mentioned how we came up with this idea and what was the motivation behind this thesis project as well as what can be the possible impact.

1.1 Overview

The whole high tech based world that we are enjoying right now is sitting on just two numbers which are 0 and 1. These 2 digits are the reason behind super computers, high tech VFX graphics, self-driving cars, space technology, high tech cameras of 1,000+ FPS (Frames per Second), AI robots, super complex algorithms, smart devices and so on.

No matter what sort of technology we are enjoying, all of these are developed by programming. Some technology took 3 days to be developed and some took over 30 years, again some are developed by one person and some are developed by multi-billion dollar companies that have employed thousands of programmers.

This shows us the importance of programming behind such massive improvement of technology and our day to day lives. Programming is one of the hardest task that mankind is ever dealing with since the invention of programmable computing system. Typing all day long to develop a functional software or app is a paramount task. Errors and Warning make this task even harder and painful.

We are hoping for a more convenient way to write codes, that's where our thesis project comes in. We named this thesis project "Vocal" which is a voice based programming app that can write codes by interacting with real human via voice based commands. As this is an Android app we chose to go with Java programming language to demonstrate our prototype.

Our thesis project prototype "Vocal" is capable of generating Java codes by interacting with human voice in natural way. One doesn't need to speak in programming language. He or she can give command just in usual way like, “create a for loop” or “set a variable” etc. Then our app will ask for the next input like what the starting of loop, what will be the type and value of increment or decrement and so on. That’s how even a kid can do programming because they don’t need to care about syntax, comma, semi-colon etc. while generating codes using “Vocal” app.

This is a very futuristic concept of programming which needs way more research to me implemented in real world for mass use. However, our thesis project prototype is able to demonstrate the basic commands like initialization, creating variables, creating arrays, creating loops, making conditions and some other primary level codes in Java language.

1.2 Motivation

The future is moving toward hands free technology which are way more convenient to use for example wireless headphones, voice assistance on our smartphones, voice based search console, podcast contents etc. This concept is getting so popular that giant companies are launching voice based assistants like Amazon Echo (by Amazon), Google Home (by Google), Home Pod (by Apple), Siri (by Apple for smartphones), Cortana (by Microsoft for PC) etc. On the top of that Google is getting over 20% (percent) mobile search that are generated by voice and this number is growing at massive scale [1].

These products and the studies of Google search are clear indication that people are in love with hands-free technologies. In future, we can see more and more such tech gadgets and assistants that will help people in day to day life just by taking commands by voice. Robots are one of a kind assistant to help us but we still need virtual assistants for non-physical tasks.

This voice based trend of technology is our main motivation behind this thesis project. We thought to make something that will be helpful for the developers using this voice based controlling system. On the top of that, our thesis project can be immensely helpful to tech programming to the kids as well as physically disabled programmers. Someone new in programming can easily learn coding using the natural language and can have in-depth idea of programming way more faster than traditional method.

Forecasting the future and developing products according to that is the most viable way to survive on this competitive world. Hence, we worked on futuristic idea that may seem nearly impossible for today but will be a daily usable product in future. We are highly motivated to solve real life problems that can improve our life styles and increase our ability of performance.

Programming is mainly popular on computer device traditionally, here comes our break through concept. Before 15 years desktop computers were our main tool to develop software and other applications. Then laptop came into the play and took the major role, right now most of the development works are done on laptop. The next generation will be focused on mobile device for almost all types of works, even now we do most of our image editing, photo retouching, content writing etc. works on our smart phones. Performance improvement rate of mobile device is increasing way faster than the performance of computers. Now a days, most of the new smartphones run on Octa-core processors where else most of the computers are running on Quad core processors. On the top of that, more and more companies are starting developing mobile chips [2].

This growth rate and user usability shows that soon people will be using mobile device for programming purpose which is one of our greatest motivations behind this work.

1.3 Project Orientation

This section of paper will simply provide the basic idea of our thesis project prototype regarding the technical issues. If you want to know the simple flow of our work then this section will be immensely helpful.

Our whole thesis project is divided into three major parts just like a computer which are taking voice based input from user, processing the input and generating the code and compiling the code. For taking the input from user we are using Google Voice API, where we don’t have much to add but the next step is our main area of work. Code compilation part is for our future implementation as it requires huge amount of storage for libraries and JDK data.

We take the voice instructions from the user which gets converted into sentence using Google API and break down the whole sentence into words. Then our algorithm detect user intention based on keywords and phrases. Later on these keywords and phrases are used to generate corresponding Java codes using our algorithm. As we have worked on limited amount of functions and syntax on this prototype, it's pretty much less complicated but the algorithm will be way more complicated after it allows to use external libraries, extended methods and other complex programming sections.

We have discussed the whole work process in detail which can be found in methodologies section of this paper.

1.4 Impact

No matter what kind of thesis project we are working on, if it has no impact on our real lives then it's worthless to invest time in. Hence, having impact is pretty necessary to calculate the success ratio of any thesis project or task. What we are working on will be beneficial for the developers and for the kids to have better idea of programming.

Right now the way developers are working just by sitting on their desks is pretty harmful for their body and health. On the top of that, typing all day long makes the hand palm painful, where else the errors and warning make it worst. Through voice based programming developers can generate codes while walking and in most cases they don't need to check for small mistakes like forgetting semi-colon, inverted comma, writing wrong syntax etc.

Apart from the developers ease, this thesis project will make programming hands-free and more comfortable at any situation. Mobile based programming will be the best way to write codes even a developer is out of his or her usual work station. It will give him fast executional scope to turn their thoughts into code in real time. Apart from that, it can be greatly useful for the testers, they can generate the chunk of codes to test it and find bugs. They don't require any laptop or powerful computer to execute the codes.

One of the main reason behind this thesis project is to tech programming in much more easier way to both newbie programmers and kids. Now a days, programming is so important that every kid must learn coding from their early age [3]. Our thesis project will be highly impactful for the kids to grab basic idea of programming without even knowing about the small complicated stuffs. It will help them to simulate their chunk of codes and learn from that.

Although there are very few physically disabled programmers who are facing problem in their career due to major accident or nerve related illness, this thesis project has a scope to help these minor amount of people to brighten their hope for life. Currently, 10% of the world's population are facing disability which is roughly 650 Million [5]. This number is astonishing and becoming liabilities for millions of families in hundreds of countries. If only 1% of these disabled people get interested in programming then it will be huge number of people who will be helpful with voice based programming. It's a great inspiration for us to bring positive change for thousands of lives using our thesis project.

Keeping above mentioned terms in mind, it's pretty clear that our thesis project Vocal can be very impactful for active programmers, disabled programmers and kids.

Chapter 2
Literature Review

From the beginning software development has been so much text-oriented and programming languages have been typewritten. In recent years, there has been an increase in the quantity of PC software engineers experiencing Repetitive Strain Injury (RSI) which has been a big concern. For those people, who are fed up with typing thousands of lines of codes, speech recognition is a good solution to reduce typing more and more codes while programming.

We are not the first who are working on voice based compiler system. A lot of works have done previously related to our work. Language processing software's are quite demandable nowadays. For this thesis project we reviewed many of the conversion topic of language processing such as voice to text conversion software and text to speech conversion software.

Stephen C. Arnold, Leo Mark, John Goldthwaite proposed a system in their paper Programming by voice, VocalProgramming [6] that a person who has RSI which is repetitive stress injuries can do programming without typing. This paper demonstrates a layout for a structure that makes circumstances that enables people to program by voice and a procedure for choosing whether the system is productive. It moreover exhibits how this generator can be used to help entering data and making XML reports.

In "Programming By Voice: A Domain-specific Application of Speech Recognition", [7] Begel tried to design the language around a natural verbalization of the code as it appears on the screen. He also tried to balance the ease of use of the language with the ability of algorithms to understand it. They conducted an experiment in which participants read a one-page pre-existing Java program out loud and found that there does exist a common vernacular among programmers for speaking programs despite the diversity of their educational training. This enables them to create a verbalized programming language definition which will work for most programmers. While doing this experiment they faced challenges such as punctuation (found in almost every programming language construct is inconsistently verbalized and is often omitted), homophones (words that sound alike but are spelled differently), capitalization of words and concatenated words. Based on these experiments and studies, Begel had developed Spoken Java, a dialect of Java that is more naturally verbalized by human developers along with a command and control language designed to enable programmers to find and select pieces of code and modify them in high-level linguistic ways. Besides to support the combination of Spoken Java and an associated command language, they built an Eclipse IDE plug-in called SPEED (for Speech Editor). This made easier to enable programmers to use voice recognition for all three programming tasks: composition, editing and navigation. Their user studies helped inform the design of Spoken Java and help improve its command language [7]. Begel did not use natural language rather he used programming language for speech recognition. He had to utter every command part by part whereas we take natural language from the use and analyze the sentence to generate the code.

In "An Investigation into Programming by Voice and Development of a Toolkit for Writing Voice-Controlled Applications", Snell developed an editor called CodeTalker which can work on any platform with any speech recognition program. The primary goal of the thesis project was a voice enabled programming editor which provides a toolkit for developing all kinds of voice controlled software applications besides support programming. The interface of CodeTalker is controlled by voice. CodeTalker supports HTML and Java where there are more support for HTML than Java. Though it's an all-in-package, there are some lacking too. One need to install a speech program and one need to compile code separately in the command prompt. [8]

In Voice language translator [9] includes a voice recognition module, a voice synthesizer, a speaker, a microphone, and a programmed central processing unit (CPU). A series of words and phrases stored in the language cartridge along with instructions to the user to speak the words or phrases as they appear. When the user speaks the words or phrases it process it and convert it to the text. This invention is directed to language translation and more particularly a voice language translator for translating words spoken in one language to spoken words in another language.

In the Real-time text-to-speech conversion system, [10] it relates to text-to-speech synthesizers and more particularly to a software-based synthesizing system which is capable of producing high-quality speech from text in real time using most of any popular 8-bit or 16-bit microcomputer with a minimum of added hardware. Richard P. Jacks, Richard P. Sprague describes in their paper that the system first compares test words to an exception dictionary and if the word is not found therein, the system applies standard pronunciation rules to the text word. Then in either instance, the text word is converted to a phoneme sequence. By the use of look-up tables addressed by pointers contained in a phoneme-and-transition matrix, the synthesizer translates the sequence of phonemes and transitions there between into sequences of small speech segments capable of being expressed in terms of repetitions of variable-length portions of short digitally stored waveforms. In general, unvoiced transitions are produced by a sequence of segments which can be concatenated in forward or reverse order to generate different transitions out of the same segments; while voiced transitions are produced by interpolating adjacent phonemes for additional memory savings. Pitch can be varied for naturalness of sound, and/or for intonation chances derived from key words and/or punctuation in the text, by truncating or extending the waveforms of individual voice periods corresponding to voiced segments."[5] In this paper , the system does real time text to speech conversion, which we did in our thesis project by Google speech cloud API and that is pretty simple and easy than this.

In VoiceGrip: A Tool for Programming by Voice[11], Alain Desilets described that programmers first utter code using an easy to pronounce pseudo-syntax and then translate that automatically to native code in the appropriate programming language. It supports the programming languages C/C++ and Perl and the editors Emacs and MS DevStudio. It is mainly a collection of macros that change text into code. Using the file extension, it translates from English-like syntax to one of the supported programming languages. In the paper there is also a feature of evaluating the performance of the system's symbol translation algorithm. In the experiment done in VoiceGrip, the system exhibited low error rates in the range of 2.7% when confusion between homophonic symbols (i.e. symbols that have the same spoken pseudo code form) was ignored and 6.6% when confusion between homophonic symbols was taken into account. Though VoiceGrip came far away in addressing the widest range of programming-by-voice problems but VoiceGrip suffers from awkward, over-stylized code entry, and the inability to exploit the structure and semantic meaning of the program.[11]

In "NaturalJava : A Natural Language interface for Programming in Java"[12], the authors describes the system of NaturalJava as they have created NaturalJava, a prototype for an intelligent, natural-language-based user interface that allows programmers to create, modify and examine Java programs. Programmers describe programs using English sentences with interface of authors and the system automatically builds and manipulates a Java abstract syntax tree (AST) in response. When the user is finished, the AST is automatically converted into Java source code. The Java code is being displayed simultaneously during the programming process. As a result, the programmer can see the code as it is being generated. The interface exploits three subsystems. The Sundance natural language processing system accepts English sentences as input and uses information extraction techniques to generate case frames representing program construction and editing directives.

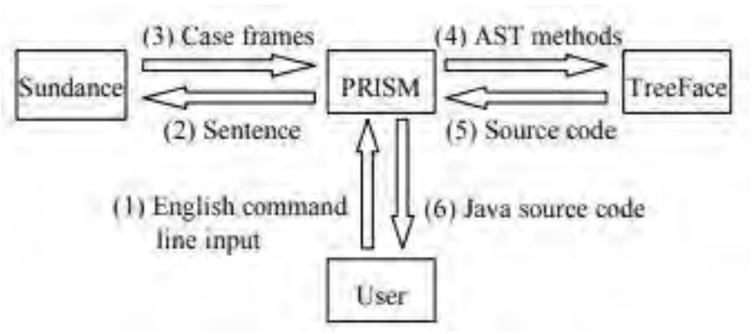


Figure 2.1: Architecture of Natural Java

This is the architecture of NaturalJava[12]. There are some limitations in this system. NaturalJava supports a large and incomplete subset of Java. It does not support array declarations because the required case frames and associated logic to Sundance and PRISM have not been added. Similarly, it does not support nested classes because the required AST support into TreeFace have not been built. The system need for compiler and debugger feedback to be coordinated with the AST interface. Moreover, it needs to generalize PRISM and add more case frames to increase the vocabulary of Sundance.

VoiceCode is a thesis project begun primarily by the National Research Council of Canada (Désilets, et al. 2006). It works with Dragon NaturallySpeaking which is a commercial speech recognition program and Emacs which is an editor. In this version of VoiceCode, programmers can code by speaking fairly English-like pseudocode into a microphone. VoiceCode had a new release in Decemebr 2006 which has extensive support for Python, C/C++ and a less support for Java, JavaScript and PHP. (Appendix 1: Online Resources). Many types of Java statements were supported by the code for "c-style languages" (C/C++, Java, JavaScript, Perl, and PHP). Another developer added a small amount of Java-specific support (define-language statement, editor-specific code for language identification, and a template for Java import statements). We found another author, Christine Masuoka who published a paper named "Java Programming Using Voice Input: Adding Java Support to VoiceCode"[13] in 2008. He added Java supports to VoiceCode and some other features in his paper. He added commands (loop templates etc.) and their spoken forms to the VoiceCode program in his paper. He kept the spoken forms for Java consistent with spoken forms in other languages. He also built in another common commands (println, main method) and set them up to do a lot of automatic typing for the user. There are some limitations in VoiceCode and two important of them are the complexity of installation and the amount of hand use involved in startup. In order to limit the amount of typing and mouse use required to start VoiceCode, Christine Masuoka has created a batch file to start Dragon NaturallySpeaking and VoiceCode. [13]

We found another paper named " A Speech Recognition based Approach for Development in C++ "[14] by Mubbashir Ayub and Muhammad Asjad Saleem where they proposed a methodology for C++ programming language where a programmer will speak and code in C++ will be written accordingly. Their system is divided into three parts- Graphical User Interface (GUI), voice to text converter and code generator. GUI gives the visual appearance of all the function to the user. Voice to text converter includes Windows Speech Recognition to take voice input and Microsoft Speechlib API which converts the voice to text. Code generator is the module which generates C++ code from listened words.[14] This system is for personal computer and pretty similar with our one.

We came across with another paper "An Empirical Evaluation of a Vocal User Interface for Programming by Voice" by Amber Wagner and Jeff Gray where they developed a VUI(Vocal User Interface) for a GUI(Graphics User Interface) based application. This app is basically developed for the disabled people who cannot use keyboard or mouse. Mapping process has been completed using MYNA.[15]

There is a major difference in all the mentioned previous works with our Vocal app. All the above mentioned works for voice to code are developed for web/computer platform whereas our vocal app is developed for mobile platform. There are many reasons behind this. Now-a-days with the advancement of technology, the world is getting closer than ever. People are getting attached to mobile phones or tablets than Personal Computers. People can carry mobile/tablets at any places they want because of light weight whereas moving PC's is quite difficult. In addition, people can afford smart phone because of its low budgets these days while we need to save a healthy amount of money to buy a computer which is not easy for all. We can notice that kids of these days get tablets for playing video games or watching cartoons. They can practice coding through our vocal app beside playing videos games and watching cartoons. As a result they can get interested in coding from an early age which will help them in future to be good coder. In all the paper we mentioned here, we saw that in all their systems they do take voice input through microphone whereas in our vocal app we take voice input through Google API in mobile devices. In our previous paper reviews we did not find any paper where they develop any voice to text system for mobile device.

Chapter 3
Methodologies

3.1 Choosing the API

As we are developing a voice based compiler application, first of all we needed to choose the better voice based API. We did research with many API's. Later, we picked three voice based API's for the final pick which are IBM Cloud API, iSpeech API and Google API.

3.1.1 IBM Cloud API

The IBM Speech to Text service provides Application Programming Interfaces (APIs) that add IBM's speech recognition capabilities to applications. The service transcribes speech from various languages and audio formats to text with low latency. The service returns and actively updates notes with the hearing of more speeches. The Speech to Text service offers three interfaces for speech recognition:

1) A WebSocket interface, for establishing persistent, full-duplex connections with the service.

2) An HTTP REST interface, that supports both session-less and session-based calls to the service.

3) An asynchronous HTTP interface, that provides non-blocking calls to the service.

The service also provides a customization interface where anyone can expand the vocabulary of a base model with domain-specific terminology or adapt a base model for the acoustic characteristics of anyone's audio. SDK's are also available that simplify using the service's interfaces in various programming languages.[16] The systems where IBM cloud API are used are embedded devices, vehicle accessories and voice control of applications, dictating email messages and notes, and transcribing meetings and conference calls etc. The Speech to Text API consists of the following groups of related calls: models, webSockets, session-less, sessions, asynchronous, custom language models, custom audio resources etc. The IBM cloud API uses HTTP basic authentication. Without embedding their service credentials in every call, applications can also use tokens to establish authenticated communications with Watson services. The accuracy rate of IBM Cloud compared to iSpeech API and Google API is 55-65%.

3.1.2 iSpeech API

The iSpeech API allows developers to implement Text-To-Speech (TTS) and Automated Voice Recognition (ASR) in any Internet-enabled application. Any device that can record or play audio that is connected to the Internet can use the iSpeech API. The API can be used with and without a software development kit (SDK). It needs internet connection, HTTP protocol, request/responses and API key. The iSpeech services cannot work without an internet connection. The iSpeech API follows the HTTP standard by using GET and POST. Requests can be in URL-encoded, JSON, or XML data formats. We can specify the output data format of responses. For TTS, binary data is usually returned if the request is successful. For speech recognition, URL-encoded text, JSON, or XML can be returned by setting the output variable. An API key is a password that is required for access. iSpeech API features are text to speech, automated speech recognition, position markers and visemes. In text to speech we can we can synthesize spoken audio through iSpeech TTS in a variety of voices, formats, bitrates, frequencies, and playback speeds. Math markup language (MathML) and Speech synthesis markup language (SSML) are also supported. In automated speech recognition we can convert spoken audio to text using a variety of languages and recognition models. And it can create custom recognition models to improve recognition quality. In position makers, we can get the position in time when words are spoken in TTS audio. In visemes we can get the position in time of mouth positions when words are spoken in TTS audio. For authentication all calls to the API require an API key as an parameter.[17] We found the accuracy rate of iSpeech API compared to IBM Cloud and Google API is 60-70%.

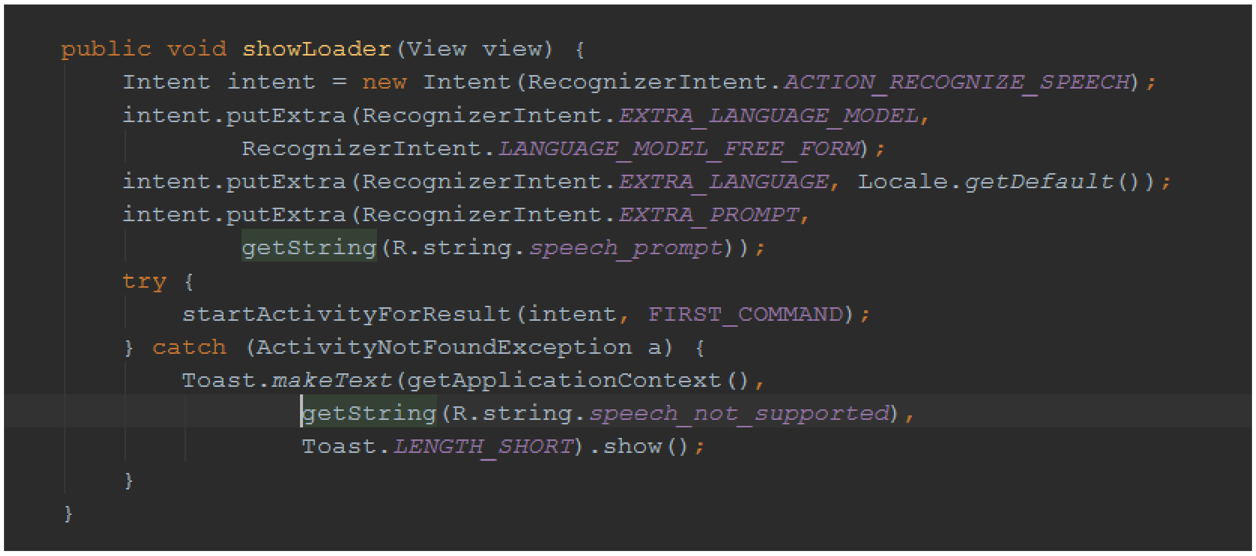
3.1.3 Google API

Google Cloud Speech API converts sound to text by applying capable neural system models in a simple to utilize API. The API recognizes over 110 languages and variants which helps to get connected with more people globally. We can also filter inappropriate content in text results. For speech recognition, it applies the most advanced neural network algorithms to our user's audio with unparalleled accuracy. Speech API accuracy improves over time as Google improves the internal speech recognition technology used by Google products. Speech API returns text results in real-time. Speech API can return recognized text from audio stored in a file. Speech recognition can be customized to setting by giving a different arrangement of word clues with every API call which is helpful particularly for device/application control use cases. Moreover, we don't need advanced signal processing or noise cancellation before sending audio to Speech API because the service can successfully handle noisy audio from a variety of environments. Last but not the least Google speech API works with apps across any device. It supports any device that can send a REST or gRPC request including phones, PCs, tablets and IoT devices (cars, TVs, speakers).[18] we found the accuracy rate of Google Cloud Speech API compared to IBM Cloud and iSpeech is 75-85%.

After doing research and testing, we found that Google API gives the most accurate result among the three API's we described here. Being the most recognized, authentic, incompatible and secured company we used Google Cloud Speech API in our Vocal app.

3.2 API Integration & Code

For API integration to our vocal application, first we need to create a Recognizer Intent by setting necessary flags such as ACTION_RECOGNIZE_SPEECH – Simply takes user's speech input and returns it to same activity LANGUAGE_MODEL_FREE_FORM – Considers input in free form English.



```java
public void showLoader(View view) {
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
            getString(R.string.speech_prompt));
    try {
        startActivityForResult(intent, FIRST_COMMAND);
    } catch (ActivityNotFoundException a) {
        Toast.makeText(getApplicationContext(),
                getString(R.string.speech_not_supported),
                Toast.LENGTH_SHORT).show();
    }
}
```

Figure 3.1: Code of API integration

3.3 Developing the Vocal app

3.3.1 Creating the Database

After choosing the API, we began to develop our vocal app in Android. Initially we used firebase database in our thesis project to store the conditions, commands and methods etc. As firebase database is an online database and it needs internet connection every time to access it, our application was getting slow. That is why, we moved from the firebase database and created our own database in the application. We loaded all our conditions, commands and methods in app database. When the user installs the application on their mobile devices the database will be automatically setup there.

3.3.2 The methodology of Vocal app

First of all the application will take voice input via Google cloud speech API from the user. Then we processed the user command and break down the whole sentence into words. As we are in the entry level of this application, we stored some keywords and phrases in our app database. Our algorithm detect user intention based on found keywords and phrases. Later on these keywords and phrases are used to generate corresponding Java codes using our algorithm. Then finally it will show outputs in the application interface. As we have worked on limited amount of functions and syntax this prototype, it's pretty much less complicated but the algorithm will be way more complicated after it allows to use external libraries, extended methods and other complex programming sections. After generating the code, it can be compiler and run via remote cloud server. As all the apps of Android OS runs on Dalvik virtual machine it's not possible to compile the code on the mobile device. On the top of this, even if we use NDK for compiling the codes inside the mobile device it will required huge space and processing power, which is not even possible on many low end mobile devices. Hence, we chose to compile the code on cloud server. In our project, we have use an open source cloud compiler server which compiles the codes and sends back the output to our app.

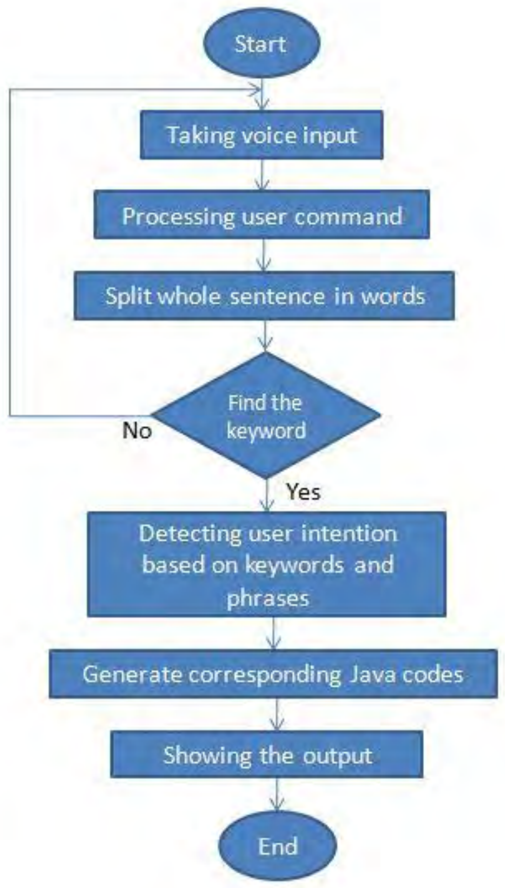

Start
Taking voice input
Processing user command
Split whole sentence in words
Find the keyword
No
Yes
Detecting user intention based on keywords and phrases
Generate corresponding Java codes
Showing the output
End

Figure 3.2 Flowchart of Vocal app

Algorithm

```
voice = input voice
array[]words = split_voice_to_words(voice)
if words contains keywords
        detect_user_intention
        generate_java_syntax
        display
        finish
else
        kill process  //for no redundancy/recursion
        restart process
```

3.3.3 Keywords and phrases

In our application database, we stored some of the keywords and phrases so that we can operate the app smoothly in our thesis project. The keywords are "print"," int", "char", "for", "while"," greater"," great", "less"," void"," return", "shutdown". And the phrases are "Create an Array", "Create a loop", "Create a class", " Greater than", " less than". We will add more keywords and phrases in our app database.

3.3.4 Different Functions

We have added many functions in our vocal application. They are variable, array, print, for/while loop, creating class, condition, methods etc. In the variable segment, we can create two types of variables, one is integer "int" and other one is character "char".

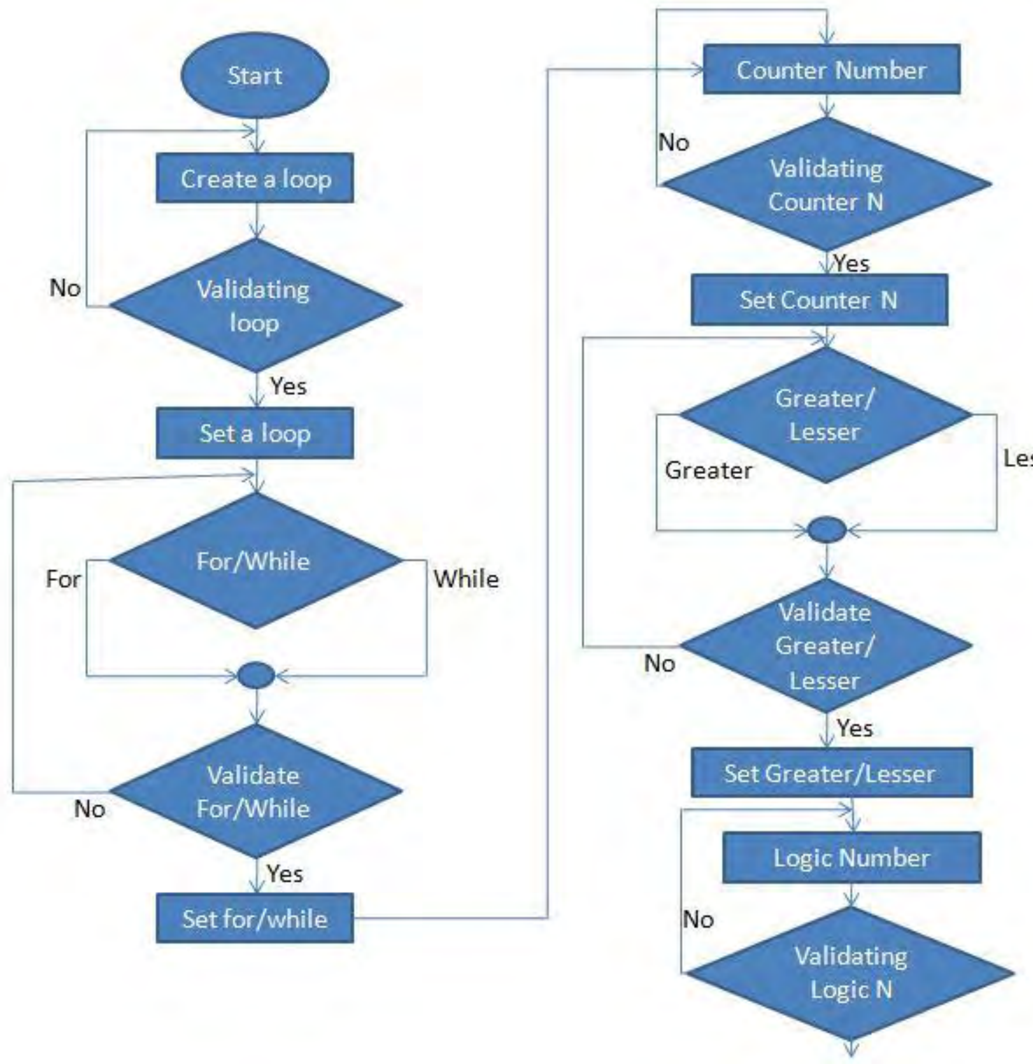
We can create one dimensional array in this app giving the command "create an array". The application will ask the size of the array. After giving the size of the array the application will create an array with the given size.

User can also print his/her result using the command "print".

We can create a class by giving the command “Create a class”. Then the application will ask for the class name. And we have to give a name for the class. By this way, a class can be created in our vocal application.

In the for/which loop there are some conditions in our application. Firstly, we will give voice input "Create a loop". It will ask "for/which" loop. If we give either for or which loop, it will ask for counter number which is how long the loop will continue. After giving the counter number, the application will ask greater than or less than. After that it will ask logic number which is the

number of increment or decrement. After that, vocal app will ask the loop will increase or decrease. Telling the increase or decrease, our vocal application finally will show the full loop.


Start
Create a loop
Validating loop
No
Yes
Set a loop
For/While
For
While
Validate For/While
No
Yes
Set for/while
Counter Number
Validating Counter N
No
Yes
Set Counter N
Greater/ Lesser
Greater
Lesser
Validate Greater/ Lesser
No
Yes
Set Greater/Lesser
Logic Number
Validating Logic N
No

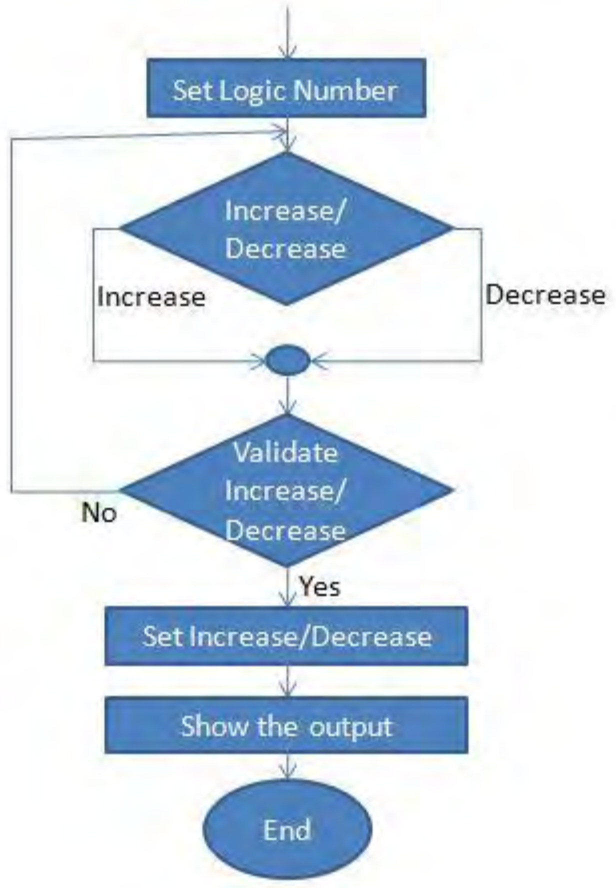


Figure 3.3: Flowchart for Loop execution

Algorithm of Loop function

```
do create loop
while validating
set loop type
do
if for/while
if validated
    set for/while
    count number
    if counter validated
    set counter number
    if greater/lesser
    validate greater/lesser
        set greater/lesser
        logic number
        if validating logic
            set logic number
            increase/decrease
            validate
            set increase/decrease
            display
```

We can also create methods in our vocal application. For this the user has to give the command "Create a Method". Then the application will ask the user what kind of method he/she wants. For the void method the user has to give command "Void" or for the non-void methods the user has to give command "Return".

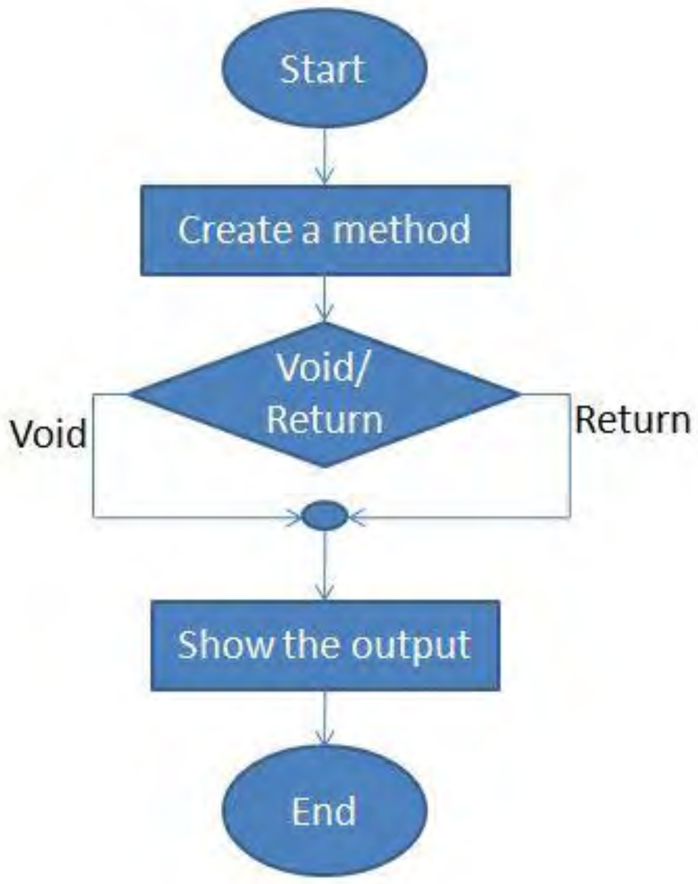


Figure3.4: Flowchart for method

Chapter 4

Experiments and Results

At present very little research has been carried out in the field of programming by voice but, as voice recognition software continues to improve and basic entry-level computers are able to cope with the computational demands of such software is increasing. In the USA alone there are in the region of 3 million people who are unable to use a computer via normal methods, but could interact with one by voice [8]. Enabling such people to programmer would open the door to the fastest growing industry in the world. So far our thesis project is on the entry level. With this level of the thesis project we did some observation.

4.1 Interacts with voice recognition system

First of all the application will take some specified commands from the user. After that the application will match the commands with its database. Then it will proceed to the next step by completing the commands. Each time the application “Vocal” take the input from the user by voice and after matching the input with its database it show the outputs in the display. In this thesis project we used Google API for taking input.

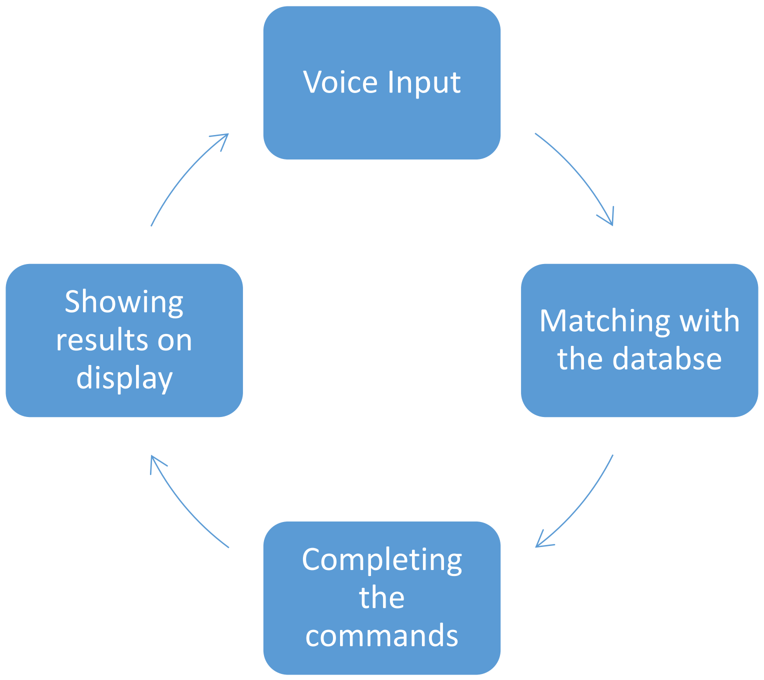
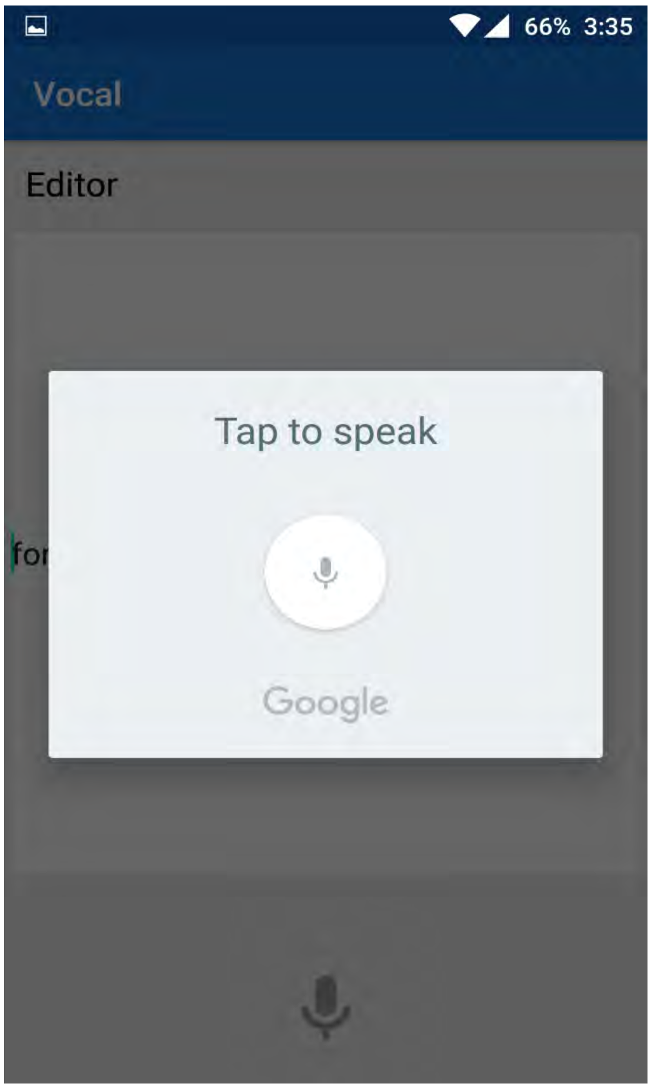


Figure 4.1: Voice recognition system

4.2 Functions

As the thesis project is at an entry level that is why we added some of the basic functions in it. After completing the full search we can add more functions and feature in the application.



66% 3:34
Vocal
Editor

Figure 4.2: Vocal's home user interface

4.2.1 Variable

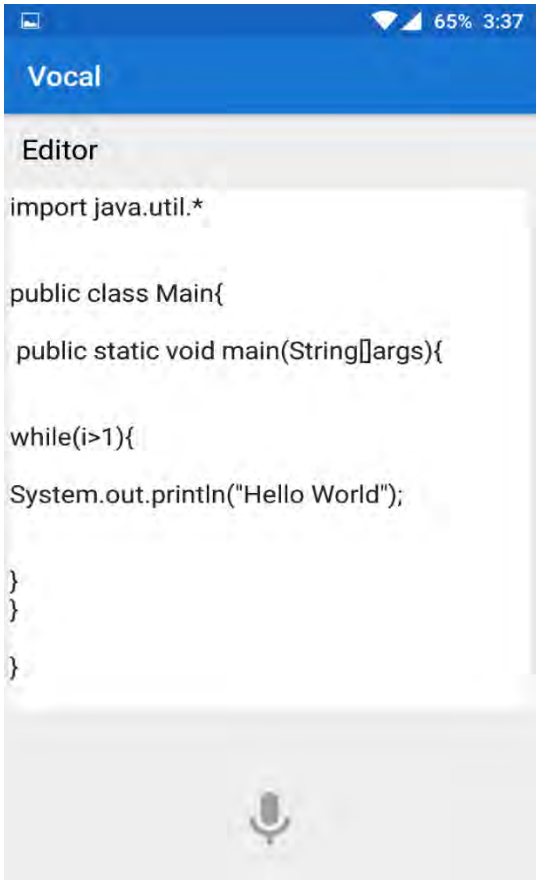
In this application, we can create variable. At this level, we only added two types of variable. One is "int" and the other one is "char". By giving commands we can create different types of variable.

4.2.2 Array

In this level we only implemented the one dimensional array. In the application "Vocal" when the user gives the command "Create an Array", the application will ask the user the size of the array. After giving the size of the array the application will create an array with the given size.

4.2.3 Print

User also can print his/her outputs in the output part of the mobile display. For this the command is “Print”. After that the application will print the result of the previous input in the mobile display.

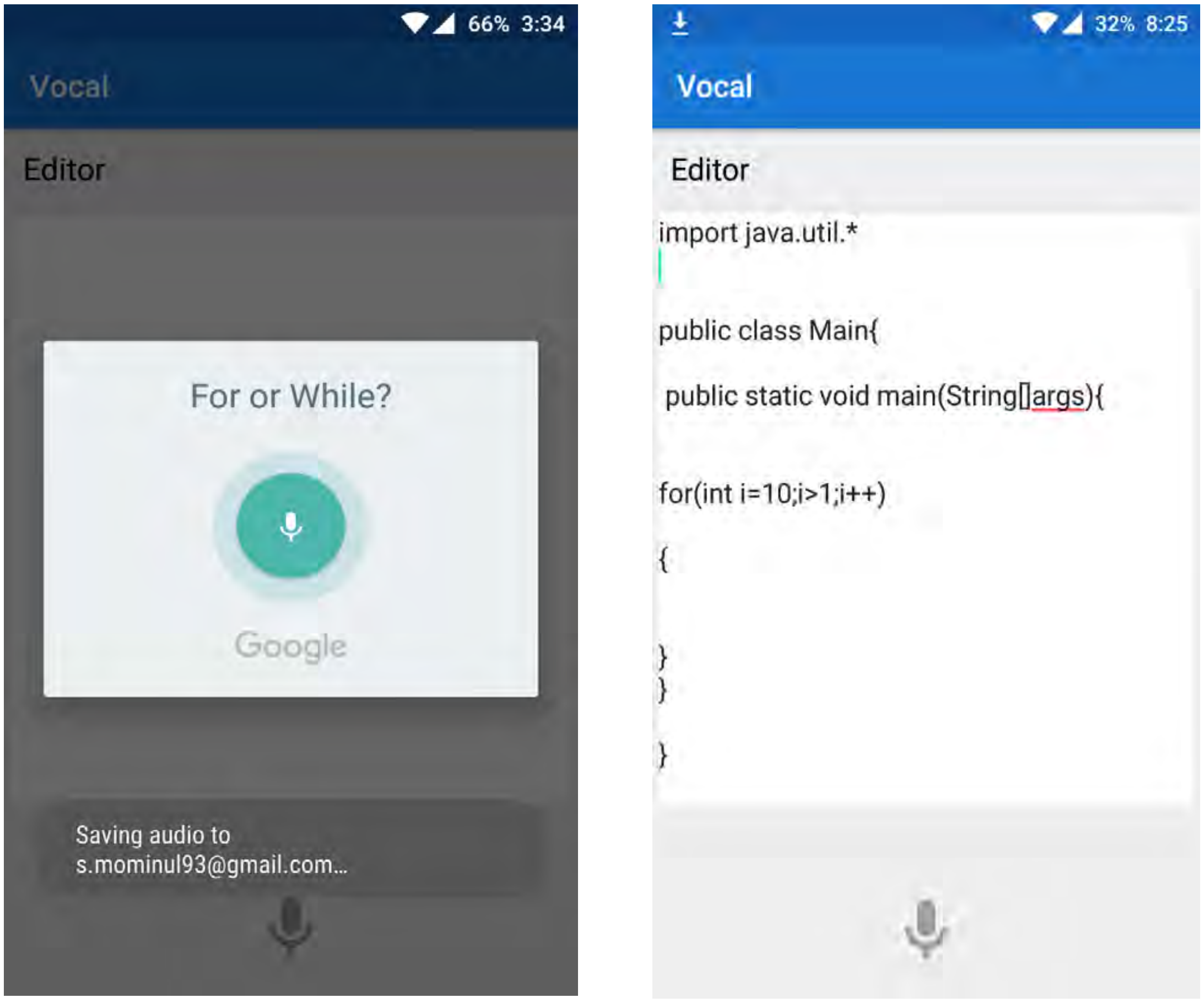


```
import java.util.*

public class Main{
 public static void main(String[]args){

while(i>1){
System.out.println("Hello World");

}
}

}
```

Figure 4.3: Vocal’s print code generation

4.2.4 For and While loop

By this application, we can create for or while loops. For this the command is “Create a loop”. Then the application will ask the user what kind of loop he/she wants. After confirmation the application will create that loop.



For or While?

Saving audio to
s.mominul93@gmail.com...

```
import java.util.*

public class Main{
 public static void main(String[]args){

for(int i=10;i>1;i++)
{

}
}

}
```

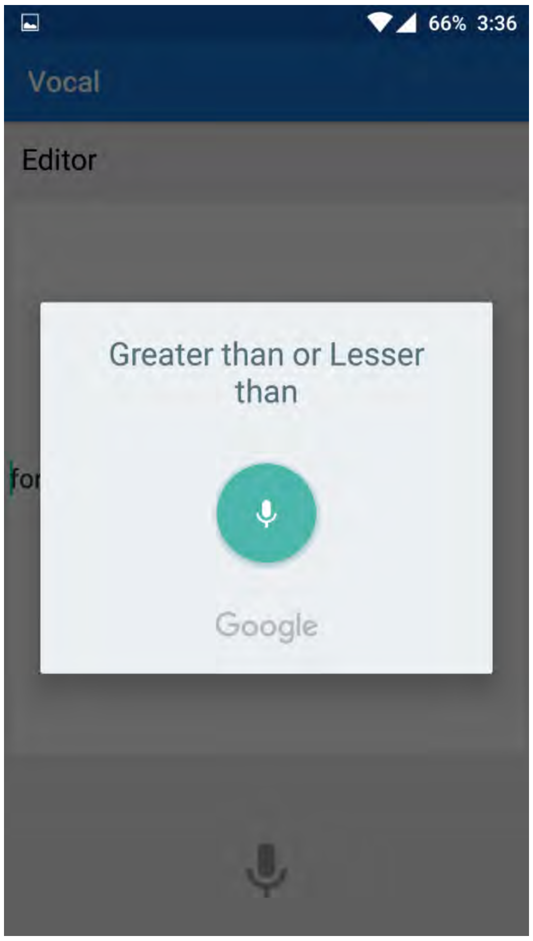
Figure 4.4: Vocal's loop code generation

4.2.5 Create a class

The user can create a class by giving the command "Create a class". Then the application will ask for the class name. Next the user has to give a name for the class. By this way, a class can be crated in the application.

4.2.6 Condition

In the application “Vocal” we added some basic conditions. For example, greater than or less than. After giving the command “Greater/Great/Greater than” or “Less/Less than” the user can choose the conditions.



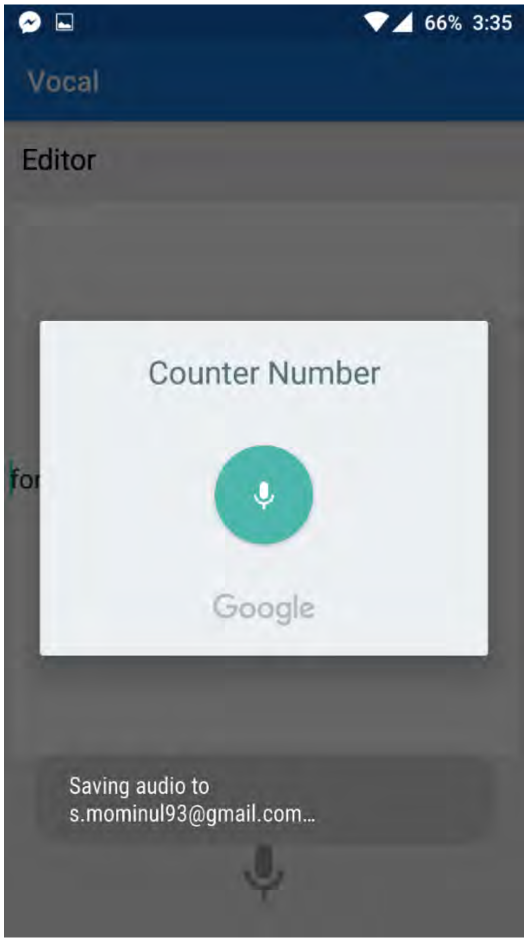


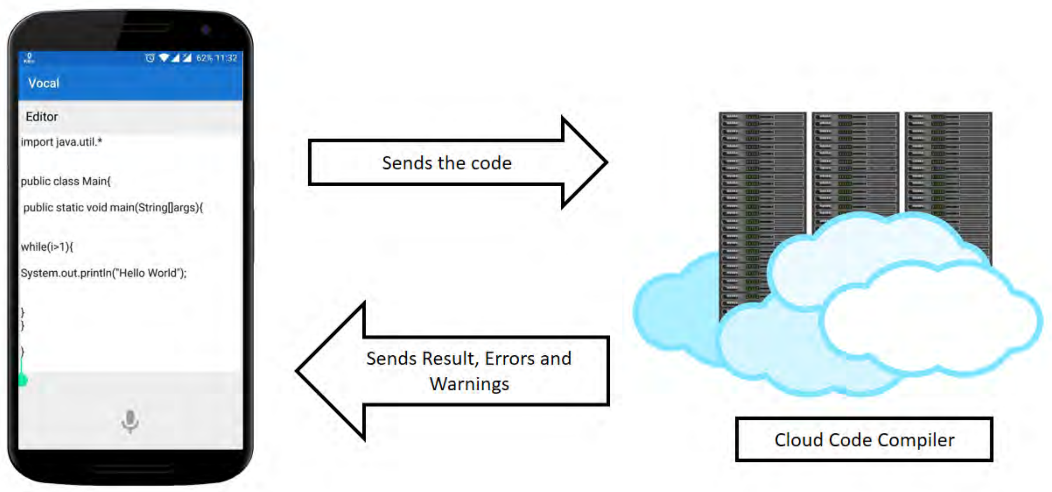
Figure 4.5: Vocal’s condition code generation

4.2.7 Methods

In our application “Vocal” we added the method function. For this the user has to give the command “Create a Method”. Then the application will ask the user what kind of method he/she wants. For the void method the user has to give command “Void” or for the non-void methods the user has to give command “Return”.

4.2.8 Compilation segment

This is an important segment in our application “Vocal”. All the outputs will be compiled on a remote cloud server. After that, cloud compiled output will be sent back to the app with Error status and even denoting the exact error. We have used cloud compiler because usually compiling on Android device is not possible because the apps run on Dalvik virtual machine and Android do not support compilation on this virtual machine.



```
import java.util.*

public class Main{
 public static void main(String[]args){

while(i>1){
System.out.println("Hello World");

}
}

}
```

Figure 4.6: Cloud compilation

4.2.9 Shutdown

Out final feature in the application is to shut down the full application by giving a command. And that is “Shutdown”. If any user gives this command the application will shut down immediately and come back to the mobile homepage.

4.3 Problems when interacting with voice recognition system

- Voice recognition software is not 100% accurate. Any software development needs to be robust enough to deal with this [8].
- Once the software has recognized a word, it will send the complete text of that word to the application regardless of what the application does. For example, any error message will be ignored until the complete command has been sent. We have to think about it when we are handling the error.
- The application fully depends on the voice input. That is why we have to test it in all the fields. Sometime a simple gap can make the full command meaning less. For example, if we give the command "Greater">>gap>>"Than" the application may not work properly. We have to give the commands properly.

4.4 How spoken forms were chosen

The mathematical precision of programming language sometime becomes curse. It is a curse for verbal entry of the programs because humans do not speak punctuation or capitalization, sometimes they drop their voice and sometime give gap in their commands. To avoid this kind of problem we experiments a lot. For example to create a loop the user can give commands in different ways, like loop/create loop/ create a loop etc. So, it may create problems in the application. That is why we choose some forms of the commands so that our application can get the right command perfectly. Here we choose the word "Loop". Whenever the application gets the "Loop" word in the command it create a loop. Same thing happens in the condition part. For example the user can give command "Great/Greater/Greater than" etc. When the application gets the word "Great" in the commands it will proceed to the "Greater than" condition.

4.5 What was not implemented and why

In the application we did not implement the “Edit” option. Because every time we give a command the cursors move to the next line. So we cannot edit the previous line of the codes. To do this we need more research in this section.

At this level of the work we did not add the “Delete” function. Because every method is related or connected with each other. If we delete one method the full code cannot be run. Same thing for the variables. If a user deletes a variable from a method but the variable is used in another method, there will be error. That is why we did not add this function.

“Copy and Paste” function is not added in the application “Vocal”. Because to copy the code we need the permission to use the phone memory. For security purpose, it is hard to access to the memory. On the other hand, if we copy a code from other places and paste the code in our applications, there may some error because it may not match with the in the application database.

4.6 Problems and solutions

In every phase of this thesis project we faced some problems. By solving those problems we moved to the next stage. Some of them are discussed below.

4.6.1 Selecting the API

To convert voice to text we need a voice based API. For this we tried many APIs. For example IBM cloud, iSpeech, Google etc. We do some research about all of this. We do many experiments with those APIs in different situations. We found this accuracy

API Name	Accuracy (%)
IBM Cloud	55-65

iSpeech	60-70
Google	75-85

Figure 4.7: Accuracy of the APIs

By this way we find out that Google API gives the most accurate result. This is why we choose to use Google API in our thesis project.

4.6.2 Creating a Database

At the beginning, we used firebase database [19] in our thesis project to store the methods, conditions, commands etc. Firebase database is an online database. That is why we need internet connecting every time to access it. But our main goal is to make the full system easy and comfortable. If the application need internet every time it will become slow and it will fully depend on the surroundings.

That is why, we moved from the firebase database and create our own database in the application. We load all our methods, conditions, commands in app database. When the user installs the application on their mobile devices the database will be automatically setup there. It will take 3-4 MB space in the mobile storage. By moving to the in app database, we solved the database problem.

4.7 Observation and analysis

To complete the thesis project we do some observation and analysis to find out the results. Some of them are given below.

4.7.1 Testing in different environments

We test our application "VOCAL" in different environments with different surroundings. For this we give this application to 10 of our university mates to test it in their class and then in their homes.

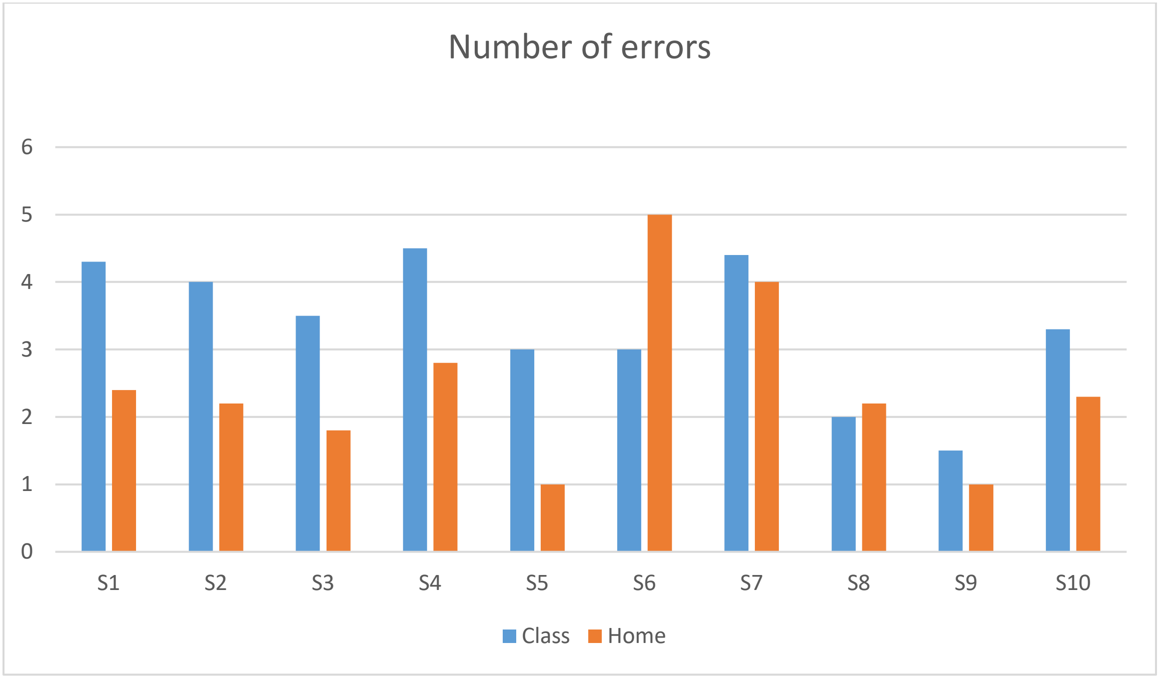


Figure 4.8: Chart of errors

From the figure (4.8) we can see that most of the errors occur in the classroom. Because we all know that the classrooms are bit noisy places. Our application fully depends on the voice. We need clear voice input to run the application properly. That is why the number of errors are more in the classrooms. On the other hand we can see that there are less errors showing in the home. The reason behind it that our houses are more silent place than our classrooms.

4.7.2 Testing with different accents

We tested our application in different accents. For this testing we used Asian accent, American accents and British accents. For the American and British accents we use voice recorder device to find out the results.

Accent	Error per person
Asian	21
American	7
British	9

Figure 4.9: Accent error per person

Here we can see most of the error occurs in Asian accent. Because Google API mainly build on the American and British accent. That is why the numbers of errors are less in those accents. We did an experiment in native English speaker and non-native English speaker.

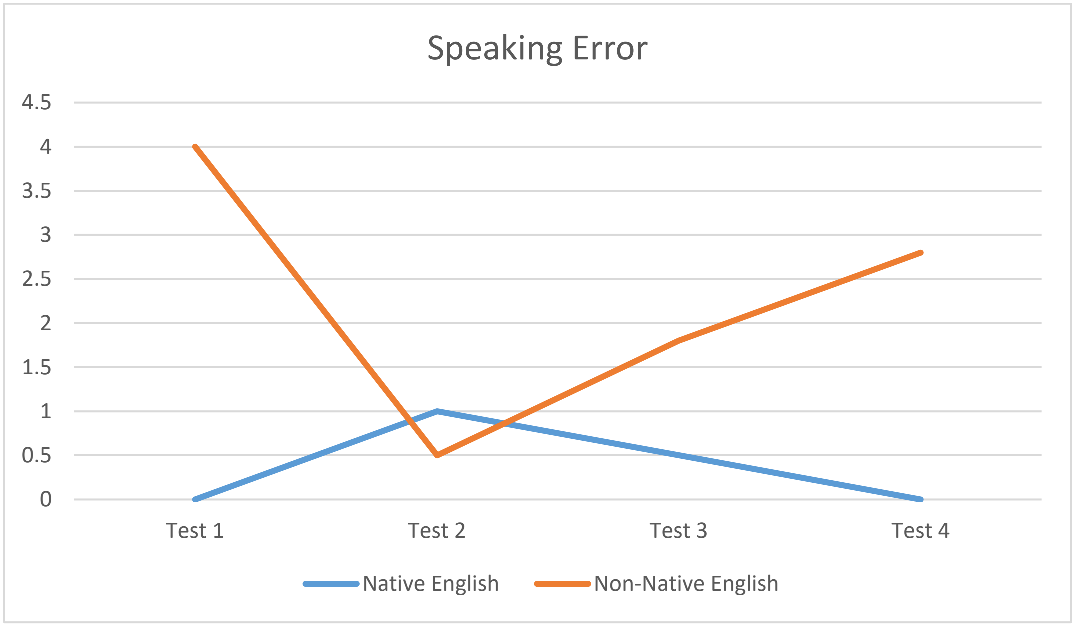


Figure 4.10: Speaking error graph

Test 2 seemed to have the less speech recognition errors for non-native English speakers, but the most speech recognition errors for native English speaker. This highlights that word pronunciation differs for each user. One native English speaker experienced zero speech recognition errors throughout all four test programs, which illustrates the accuracy of the speech recognition is based on articulation and pronunciation.

Chapter 5
Future Plan and Conclusion

Future is committed to no one hence there is no limit of growth and prosperity. Every successful product, service or company must learn to adopt and evolve to be fit for future market and demand. Same thing goes for our thesis project prototype Vocal.

Right now it's a prototype which is not suitable to be launched in market at this moment. Thus this thesis project needs way more improvement and finalization before being fit for the end consumers. Apart from the consumer product, it can play a great role in research mostly for natural programming language.

5.1 Future Implementation and Extension

There is no end of development while working on programming related thesis projects. The one we worked on is just the prototype and needs huge amount of work and future implementation to make it complete or even useable for end users.

We will be moving for more natural language detection for perfect code generation. Right now, we are dependent on Google voice API for getting input from user but it has noticeable amount of error rate when it comes to detecting non-native English speaker. That's where we need way more development for our thesis project to be market fit all over the globe. According to the survey of HackerRank there is not even a single native speaking country within top 10 [4] in the list of best developer's country, that simply shows non-native speakers are way more important when it comes to programming. We will either search for more advanced voice API that can detect all types of English accent or we need to develop one for ourselves by giving extra priority to non-native speaker's accent.

On the top of that, we can make it act like an AI by integrating neural network instead of keywords and phrases based algorithm. That's how our thesis project will reach at its maximum potential of being the most advance natural language processor and code generator that can execute based on human voice input.

5.2 Business Perspective

Every product needs to have a business value to sustain and evolve. There are basically two types of business which are mostly seen in our economy which are B2B (Business to Business) and B2C (Business to Consumer). There are very few product lines in the world which are perfect for both type of market and luckily Vocal is one of those.

There are lots of paid compiler like IntelliJ IDEA, Codenvy, Sublime Text and so on which simply shows that there is a big market for paid IDEs despite having lots of free one. Software firms and development companies mostly choose the paid IDEs due to having dedicated support while they stuck. On the top of that, sometime companies may need proprietorship extensions for their business and only such big IDEs owners can provide such service. Hence we can clearly, observe a big B2B market for our Vocal which can reduce programmers day to day pain of sitting at one place and typing codes.

At the same time, now a days smarts parents are aware how important programming is and for this reason they want their children to learn basic programming at very early age. There are already many apps, games and even hardware based tool-kits to teach programming to kids. Our vocal app can bring new horizon to this sector by opening new way of learning programming by giving voice based instructions.

5.3 Research Sector

Every single technology has its own limits and requires huge research and development to overcome the limits. In some cases, research for one particular issue may come up with solutions that can be applied in many other sectors or industries. Since the invention of programming language people have been doing research and as a result we got advanced programing languages like C++, Java, Python, JavaScript and so on. Same sort of research is need to make this voice based code generator way more efficient, faster and accessible.

Apart from voice input, its processing section has huge research scope where AI and machine learning can be implemented for more accurate identification of voice instruction.

5.4 Conclusion

Actually there is no end of tech based solution, there is nothing called 100% perfect solution of any problem because every single product, service, research thesis project etc. has more rooms to grow and expend.

Our thesis project is a small prototype of what future programming will look like and what can be done to improve this thesis project even further. Our main work relies in the instructions processing part and will be glad to work on the cloud compiling section in future.

Programming is always a paramount task for human being but voice based programing can make it way easier and simple to code. Even business minded people or other non-engineering people can do programming with the help of voice based programming method. There was always a gap between 100% natural human language and programming language, our thesis project can be the starting of dispelling this gap and make programming even more fun.

References

1. G. S. (2016, May 24). Google says 20 percent of mobile queries are voice searches. Retrieved February 15, 2018, from https://searchengineland.com/google-reveals-20-percent-queries-voice-queries-249917
2. Norris, J. (2014, February 13). Android processors: The past, present and future of smartphone chip design. Retrieved February 15, 2018, from https://www.greenbot.com/article/2095485/android-processors-the-past-present-and-future-of-smartphone-chip-design.html
3. Crow, D. (2014, February 07). Why every child should learn to code. Retrieved February 20, 2018, from https://www.theguardian.com/technology/2014/feb/07/year-of-code-dan-crow-songkick
4. Trikha, R. (2017, July 14). Which Country Would Win in the Programming Olympics? Retrieved February 20, 2018, from https://blog.hackerrank.com/which-country-would-win-in-the-programming-olympics/
5. Disability Statistics: Information, Charts, Graphs and Tables. (2018, March 16). Retrieved February 25, 2018, from https://www.disabled-world.com/disability/statistics/
6. Arnold, Stephen, Mark, Leo and Goldthwaite, John. Programming By Voice, Vocal Programming. In the Proceedings of the ACM 2000 Conference on Assistive Technologies. November 2000.
7. Begel, A. Programming By Voice: A Domain-specific Application of Speech Recognition. AVIOS Speech Technology Symposium–SpeechTek West (2005).
8. Snell, Lindsey. An Investigation Into Programming By Voice and Development of a Toolkit for Writing Voice-Controlled Applications. M.Eng. Report. Imperial College of Science, Technology and Medicine, London. June, 2000.
9. Voice Language Translator. Publication number US4984177 A. Publication type Grant.
10. Real-time text-to-speech conversion system. Publication number US4692941 A. Publication type Grand. Application number US 06/598,882
11. Desilets, Alain. VoiceGrip: A Tool for Programming by Voice. International Journal of Speech Technology, 4(2): 103-116. June 2001. Price

12. D. et al. NaturalJava : A Natural Language interface for Programming in Java. In Proc. of the Int. conf. on Intelligent User Interface (2000).
13. Masuoka, Christine. Java Programming Using Voice Input: Adding Java Support to VoiceCode . University of Maryland at College Park
14. Ayub, Mubbashir and Asjad, Muhammad. A Speech Recognition based Approach for Development in C++.
15. Wagner, Amber and Gray, Jeff . An Empirical Evaluation of a Vocal User Interface for Programming by Voice.
16. Speech to Text basic example updated · watson-developer ... (n.d.). Retrieved March 18, 2018, from https://www.bing.com/cr?IG=CA894C9FFDC24300B3B71B93BA60C425&CID=0B2185A1633E6FA7169E8E6662916ECF&rd=1&h=Y3YUlHL83QLBmPMoV7AZd5RIYZc949T1lPYx0D6KJ1s&v=1&r=https://github.com/watson-developer-cloud/node-red-labs/commit/acf066f262c81c36d6927d5fc38521de4fbe4bf4&p=DevEx.LB.1,5072.1
17. API Reference - iSpeech.org. (n.d.). Retrieved March 20, 2018, from http://www.bing.com/cr?IG=078B26B7659F4097AFE22C609E251353&CID=2EF2A4D4101767793525AF1311B8666D&rd=1&h=x9E41uC2_uVg4GNCod1A5JpPoE8mDUSDO4D-9j0iB9g&v=1&r=http://www.ispeech.org/api/&p=DevEx.LB.1,5070.1
18. Speech API - Speech Recognition | Google Cloud. (n.d.). Retrieved March 23, 2018, from https://www.bing.com/cr?IG=7918FE4B6D4541DAAC08072C7298318F&CID=067C29EA79DC6555278F222D7873646E&rd=1&h=h0gGmapXw3Q-E9iKFGvcjre02FUfkVQAXydQLiqCW1g&v=1&r=https://cloud.google.com/speech/&p=DevEx.LB.1,5503.1
19. A. S. (2017, April 24). An introduction to Firebase - the easiest way to build powerful, cloud-enabled Android apps. Retrieved March 23, 2018, from https://www.androidauthority.com/introduction-to-firebase-765262/