IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102

Programming using Voice for Physically Challenged Individuals

Adnan Rahim Khan[1],Charith C Shetty[2],Deepak C A[3],Deepanshu Kumar Pali[4],Abhinav R B[5]

Student [1DS18CS010], Dayananda Sagar College of Engineering, Bangalore-78[1]

Student[1DS18CS037], Dayananda Sagar College of Engineering, Bangalore-78[2]

Student[1DS18CS041], Dayananda Sagar College of Engineering, Bangalore-78[3]

Student[1DS18CS042], Dayananda Sagar College of Engineering, Bangalore-78[4]

Professor [CSE], Dayananda Sagar College of Engineering, Bangalore-78[5]

Abstract: Programming today demands to be typed manually as well as with the option to write code literarily. As a result, both of these have placed people with disabilities in an unquestionably difficult situation when it comes to understanding and using coding. Few of the technologies that enable computer use by people with disabilities are the specific solutions for helping people with disabilities with various coding issues.

The goal of this project is to build a portable Web interface, especially for people with disabilities to learn how to code. A web interface is planned, with a focus on speech recognition, semantic analysis, and people working together with coding tools to help physically and visibly disabled people. It is a strategy for implementing a superior method of aiding and persuading handicapped people, including the most recent developments in information innovation and computer programming viewpoints. Here, rather than adopting the more current syntax that software professionals use, we are concentrating on the C language because it is seen as being fundamental. This project helps all people with physical disabilities who are interested in coding by demonstrating programming concepts and motivating people to use programming models to develop their learning capacity and coding hypothesis. The individual is led step-by-step by their abilities, using oral orders to act immediately at any time.

Keywords: Vocal Programming , Programming tool , Speech Recognition ,Multimodal Interfaces , Voice I/O , Voice Accessibility

I. INTRODUCTION

1.1 Motivation

Literary programming has gotten its practical structure since the 1950s. When programming for the physically handicapped individuals first became a possibility, differently abled software engineers were unable to complete the task. Developers who are physically challenged have had very successful careers. In the present world, there are more employed physically handicapped software developers, demonstrating that people with upper body impairments are interested in the subject of programming.

It became harder for the disable people to find employment as software developers after the decade of 1980 with the development of the Graphical User Interface. It was possible to represent visual elements through text documents during the early Graphical User Interface eras, but as these tools have improved, this has essentially become impossible. Software programmers who are handicapped now have a huge problem to solve. The tools for assisting customers who were physically unable to programme relied on typing and code conversion. Unfortunately, these instruments haven't advanced as quickly as GUI technology, and it's been extremely difficult to rework them so they can operate under new development conditions and new programming paradigms. Clients who are physically disabled can quickly take up programming, which can dramatically improve their quality of life. If there is an application for clients who are visibly disabled, which converts client voice commands into programming structures and keywords, as well as a client strong clever intuitive, who collaborates with client and guides the client, to address this issue, such a framework demonstrates an incredible worth due to their knowledge and convenience. Additionally, such a programme will be a very helpful and adequate solution for people who are physically challenged and find it difficult to type a text on computers. The "Vocal Programmer" project focuses on developing a number of options, innovations, and programme elements that are used to learn programming language concepts and make available for better, quicker, and more accurate results for people who are physically unable. Existing programming learning resources are scarce, insufficient to give a programme voice, and inapplicable everywhere. This framework has been shown to be a two-way learning framework in this evaluation. These frameworks mostly focused on accuracy level. Many frameworks used the discourse acknowledgement API handling

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102

method when interacting with the voice acknowledgment component. Web-kit is used to interpret the message for the discourse acknowledgment component by creating their own discourse acknowledgment module. Using Google Speech API is necessary. It has emerged as a more common invention in programmed speech recognition, and many automatic speech recognition frameworks incorporate it. Developing a framework to resolve this is much more difficult than developing another discourse to text processor. The product must be straightforward to use, easy to communicate with, and most importantly, exact. The "Vocal Programmer" web application is a simple and lightweight programming tool that is helpful to use anytime and wherever.

1.2 Natural Language Processing

Software engineering's Natural Language Processing (NLP) subject is specifically concerned with giving computers the ability to understand text and spoken language in a manner comparable to how humans do.

NLP combines factual, AI, and in-depth learning models with computational phonetics, which displays human language using rules. These developments provide PCs the ability to handle human language as message or voice information and to fully "comprehend" its importance, including the speaker's or author's intention and viewpoint.

NLP powers computer programmes that interpret text from one language to the next, respond to spoken commands, and summarise massive amounts of text swiftly and even progressively. The complexity of human language makes it extremely difficult to write programming that accurately determines the expected importance of text or speech information. Homonyms, homophones, sayings, illustrations, syntax and use exemptions, and different sentence structures are just a few examples of the inconsistencies in human language that take people a long time to learn, but that programmers must teach regular language-driven applications to understand and see precisely all along if those applications are to be useful.

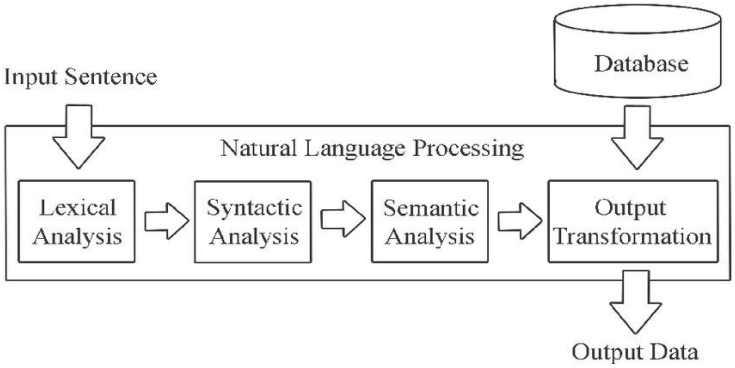


Figure 1.2: Working of a Neural Network

In order to help the individuals understand the text and speech data it is absorbing, several NLP activities deconstruct human text and voice data. These are only a few of these jobs:

The process of accurately translating voice data into text is known as speech recognition, commonly referred to as speech-to-text. Any programme that responds to voice commands or questions must use speech recognition. The way individuals speak—quickly, slurring words together, with varied emphasis and intonation, in various dialects, and frequently using improper grammar—makes speech recognition particularly difficult

The act of identifying a word's part of speech based on its use and context is known as part of speech tagging, also known as grammatical tagging.

Choosing a word's meaning from among its potential meanings by employing semantic analysis to determine which word makes the most sense in the given context is known as word sense disambiguation. By way of example, word sense disambiguation helps to distinguish between the meanings of the verbs "make," "make the grade," (accomplish), and "make a bet" (place).

By employing named entity recognition, or NEM, words or phrases are recognised as useful entities. NEM recognises "Kentucky" as a location or "Fred" as a person's name.

Co-reference resolution is the process of establishing whether and when two words refer to the same thing. Figuring out what or who a given pronoun refers to (for instance, "she" = "Mary") is the most prevalent example, but it might also call for understanding a metaphor or idiom that is utilised in the text (e.g., when "bear" refers to a big, hairy person rather than an animal).

Sentiment analysis scans texts for elusive aspects including attitudes, sentiments, sarcasm, confusion, and mistrust.

The antithesis of voice recognition or speech-to-text, natural language generation is the process of translating structured data into human language

.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022
DOI: 10.17148/IJARCCE.2022.116102

1.3 Real World-Application

This "Vocal Programmer" have numerous application and some of them are mentioned below. There are copious amount of physically handicapped and blind users available in the world so it comes to no surprise that the "Vocal Programmer" can play a vital role in here. This can be a web application which is portable to convert voice to code. The main application can be divided into two features: a web use of this application or the offline approach.

Real World use of Vocal Programmer

1. People with upper body impairments
2. Software Industry
3. Robotics
4. Machine Translation
5. Virtual agents and chatbots
6. Spam detection
7. Social Media Sentiment analysis

II. PROBLEM STATEMENT AND PROPOSED SOLUTION

2.1 Problem Statement

To develop an IDE that is used to efficiently write programs with your voice using natural language processing and speech recognition.

2.2 Existing Systems

2.2.1 Serenade.ai

Serenade allows us to create code using natural voice. Serenade's speech-to-code engine is built from the ground up for developers. Serenade can run in the cloud to reduce the effect on your system's resources, or totally locally so that all of your voice commands and source code remain on the device. Serenade interacts with our existing tools, from VS Code to any other programming tools, so we don't have to learn an entirely new process. Serenade also matches the correct speech engine to what we are editing, whether it's code. or any other sentences. Using Serenade's open protocol, we create powerful custom voice commands and plugins that we can integrate into our workflow. It is very helpful for generally typing but the main problem comes when we have to pay for some of the features and it is not an IDE for coding.

2.2.2 Dragon Professional

Dragon professional provides a speech recognition system hosted on cloud for the commercial as well as public usage that can save your time. Dragon Professional Anywhere seamlessly integrates into processes, allowing you to produce high-quality code and documentation with your voice. This software is developed by a company called Nuance. Dragon Professional helps professionals that are busy, along with the workers that work remotely, to naturally use their voice to rapidly and easily write more complete and accurate documentation. There are no per- user limits, so you can speak freely and as much as you want. Business workers may work from anywhere with ease and focus on the clients and business more rather than technology because of the time saved. With no training required of any kind and a single profile based on the cloud, that is automatically created on first use, you can get up to 99 percent accuracy. Manual tasks like accent adjustments and microphone calibration are now automated, resulting in improved precision, a lower word mistake rate, and an optimal user experience from the outset. Hence we understand that this is only a plugin used for increasing the productivity and it does not rely much on coding.

2.2.3 Voice Finger 3.0

Voice Finger is a speech recognition tool that allows you to control your mouse and keyboard with your voice as quickly as possible. Allows for zero computer touch, eliminating the need for keyboards and mice. Rest your hands and operate the computer with your voice. A permanent solution for individuals with disabilities and/or computer damage. For some jobs, some speech recognition software thinks you can type and click. Voice Finger was designed to do everything using your voice. Also suitable for die-hard gamers. Voice Finger may touch keys and buttons while the gamer moves and shoots, acting as a third hand for competitive gamers. Complete command over the mouse Other speech recognition software only supports standard mouse clicks but Voice Finger enables Double-click the mouse or use the left, centre, or right buttons .Use any of the three mouse buttons to drag and drop. Perform any of the commands listed above while holding down special keys such as Control, Shift, and Alt. Automatically clicking several times. For example, automatically clicking 30 times in a picture gallery with a 10-second delay between each image. And almost anything else you can think of. This is a tool by which you can control your computer with your voice i.e you can use your mouse and keyboard for doing small tasks, but while programming which requires to add some syntax is difficult.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102

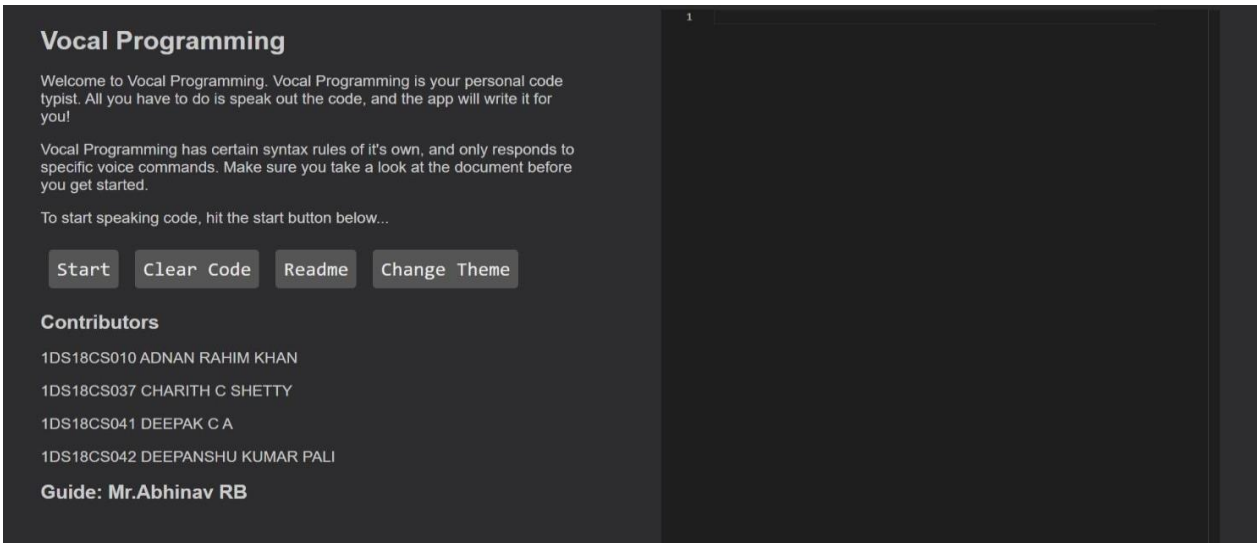
2.3 Proposed Solution

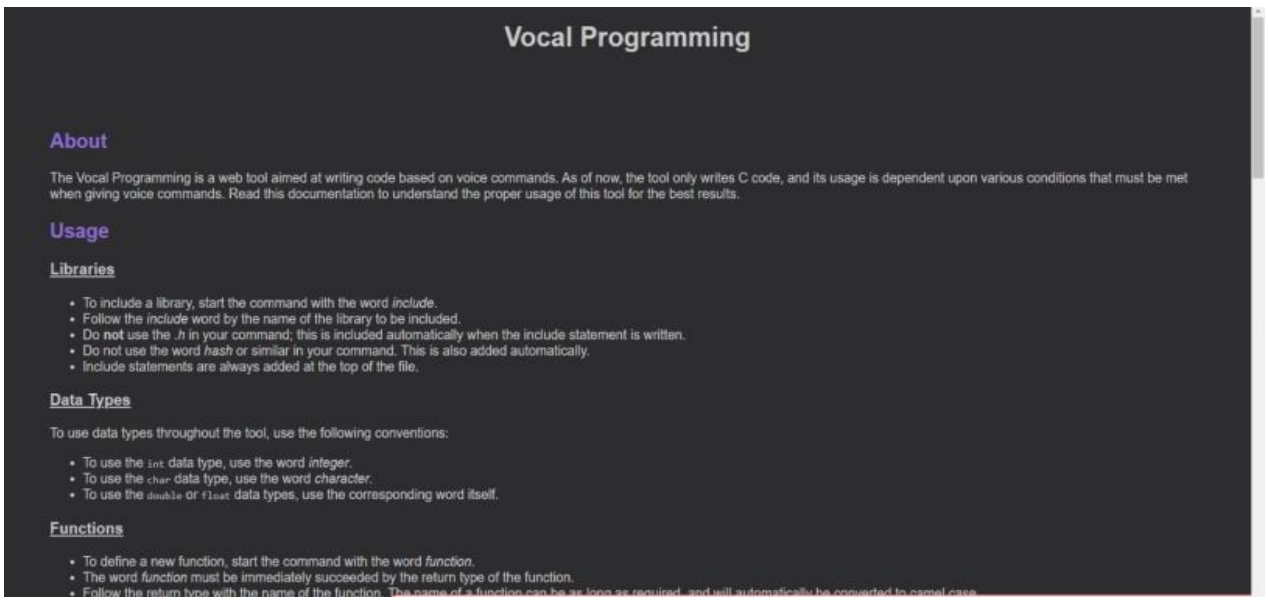
2.3.1 Vocal Programming

Vocal Programming is a web based IDE for C programming. It can be used to type C programs through voice commands. Vocal Programming is an online application that allows you to write code using voice instructions. As of present, the tool only writes C code, and its use is based on a number of requirements that must be followed when issuing voice instructions.

The screenshot below shows the main page of the vocal programming software which has different buttons, which include the buttons to perform certain functions such as start writing the code with voice commands, stop listening, clear code, change theme which changes the colour of the application and Readme which shows the different features of the applications.

We can also see the syntax with which the user has to give voice commands so that the computer recognizes the commands and generates the corresponding code. There are also some examples given in the readme page.





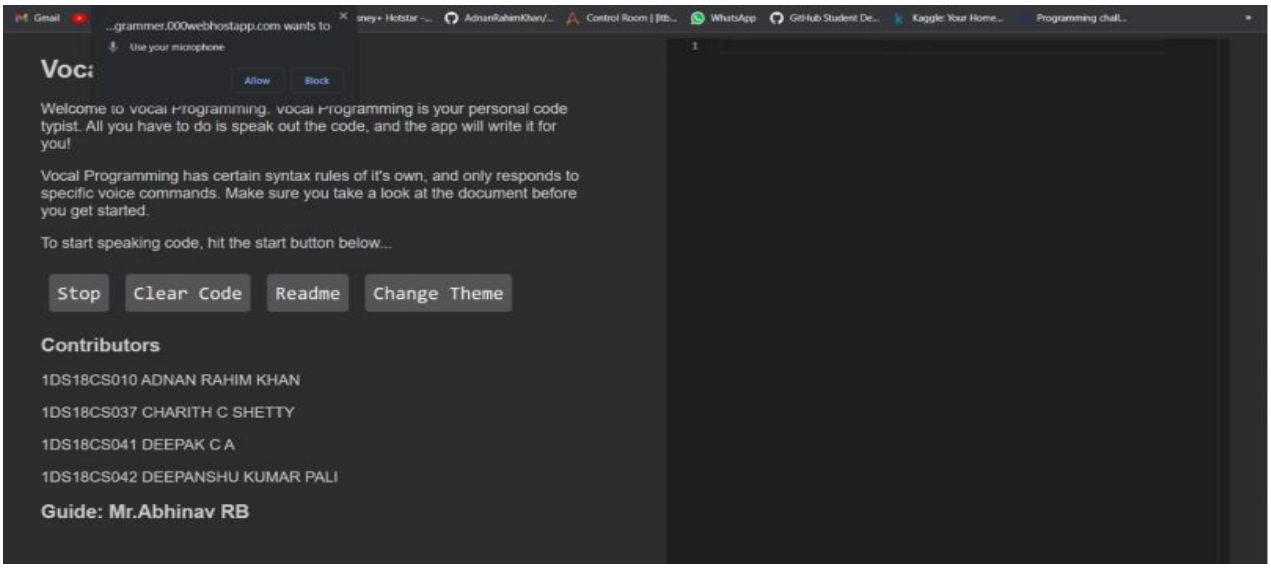
The above screenshot shows some of the commands that the application can perform tasks on. When the user has to start using the application, the user should click on the start button and the application asks for the permission to use the systems microphone. The user should click on the allow button in order to use the application, which has been shown on the screen. The user can even use the normal ways of typing i.e by using the keyboard and mouse to edit the programs, because we have used the monace editor in code editor section and added some voice instructions with it to help the user to type through voice commands.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102



2.4 Usage of Vocal Programming

2.4.1 Libraries:

To use the libraries in C language, use the steps below

- Begin the command with the word include to include a library.
- The name of the library to be added should come after the include word.
- Do not include the ".h" file in your command, it is automatically included when the include statement is created.
- In your command, do not mention the phrase hash or anything else, this is also automatically included.
- Include statements are always placed at the beginning of a file even if the cursor is somewhere in the middle or end.

Example:

Voice command: include stdio Function written: #include<stdio.h>

2.4.2 Data Types:

To include the data types in your code, use the following steps:

- To use the int data type, use the word integer.
- To use the char data type, use the word character.
- To use the double or float data types, use the corresponding word itself.

Example:
Voice command: integer var
Function written: int var;

2.4.3 Functions:

To include the functions in your code, use the following steps:

- To define a new function, start the command with the word function.
- The word function must be immediately succeeded by the return type of the function.
- Follow the return type with the name of the function. The name of a function can be as long as required, and will automatically be converted to camel case.
- To pass parameters to the function, use the word takes after the name of the function. This word must then be succeeded by the return type of each parameter and the parameter name.

Example:

Voice command: function integer average takes integer 'size' Function Written: int average(int size) {}

2.4.4 Variables:

To include the variables in your code, use the following steps:

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022
DOI: 10.17148/IJARCCE.2022.116102

- To declare a variable, start the voice command with the data type of the variable. (Remember to use the above given conventions for data types)
- Follow the data type with the name of the variable. The name of the variable can be as long as required, and will automatically be converted to camel case.
- However, you may only declare one variable per line.
- To initialize a variable while declaring it, follow the name of the variable with the word equals, then give it the initial value.
- Characters will automatically be initialized within single quotes.

Example-1:

Voice command: integer size equals 5
Statement Written: int size = 5;

Example-2

Voice command: character test equals p
Statement Written: char test = 'p';

2.4.5 Printf:

To print any message or a value using printf statement, use the following steps:

- To use the printf function, start the command with the word print.
- Follow the word with the quote to print.
- To use a variable in the quote, use the word percent/percentage in the quote. No need to use the identifier, such as 'd'; identifiers will be added automatically based on the variable's data type.
- To specify the variable to be used instead of the placeholder, use the word variable.
- Following the word variable, say the name of the variable to be used.
- To add a newline(\n)in the print quote, use the word line.

Example:

Voice command: print the value of test is percent line variable test
Statement written: printf("the value of test is %d\n ", test);

2.4.6 Scanf:

To take inputs from the user using the scanf command, use the following steps:

- To use the scanf function, start the command with the word scan.
- Follow the word with the names of variables to be scanned. Format identifiers or a quote for scanf is not required. The quote will be written automatically based on the data types of variables.
- The word ampersand/and is not required before each variable either. The '&' symbol is added automatically before each variable.

Example:

Voice command: scanf test, size
Statement Written: scanf("%f%d", &test, &size);

2.4.7 If-Else Statements:

To use the if else statements, use the following steps:

- To use the if-else construct, start the command with the word if or else.
- Follow the word with the condition. Preceed variable names with the word variable, and say constants (integers or chars) directly
- To add relational operators, use the words equals, greater, greater equals, less or less equals.

Example:

Voice command: if variable test greater equals 5
Statement Written: if (test >= 5) {}

2.4.8 While Loop:

To use the while loop, use the following steps:

- To use the while loop construct, start the command with the word while.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022
DOI: 10.17148/IJARCCE.2022.116102

- Follow the word with the condition. Preceed variable names with the word variable, and say constants (integers or chars) directly
- To add relational operators, use the words equals, greater, greater equals, less or less equals.

Example:

Voice command: while variable test greater equals 5
Statement Written: while (test >= 5) {}

2.4.9 Out:
To get out of a loop or any other statement, just say the word out. When the word out is recognized by the system, it automatically gets out of the statement.

2.5 System Requirements:
The requirements for developing and deploying our project are stated below:

2.5.1 Hardware Requirements:

1. A computer system of the following minimum specifications:

Processor	Intel Core i5-8300H CPU @2.30 GHz
RAM	4.0 GB
Operating System	Windows/MacOS/Linux
Hard Disk	512 GB

2. Internet Connection
3. Microphone

2.5.2 Software Requirements:

1. Visual Studio Code
2. Web Development Tools (HTML ,CSS ,Javascript ,Bootstrap)
3. Monaco Editor

III. ARCHITECTURE AND DESIGN

3.1 System Overview

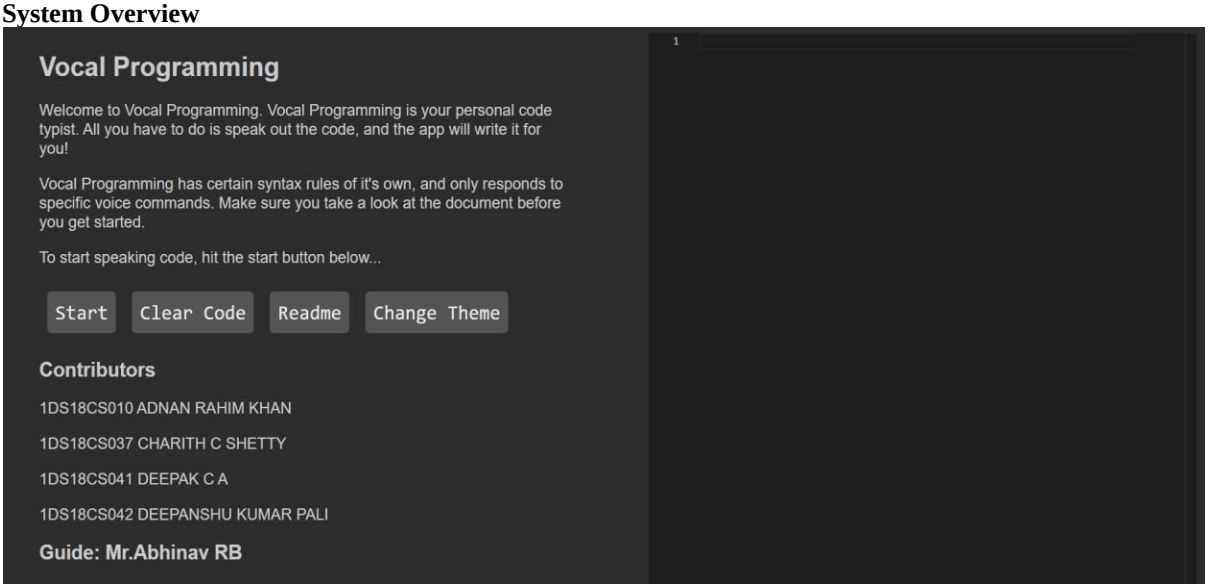


Figure 3.1: System Overview

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102

An overview of the system is shown in Figure 4.1. The related modules are displayed in the system structure, i.e

1. Start :- Inorder to start the voice recognition function.
2. Clear Code :- Inorder to clear the entir IDE Section
3. Readme :- To read the syntax or user manual for using this software
4. Change Theme :- To convert theme from Dark to Light or vice-versa.
5. IDE :- This is the section where coded output will be obtained.

3.2 Software Architecture

3.2.1 System Block Diagram

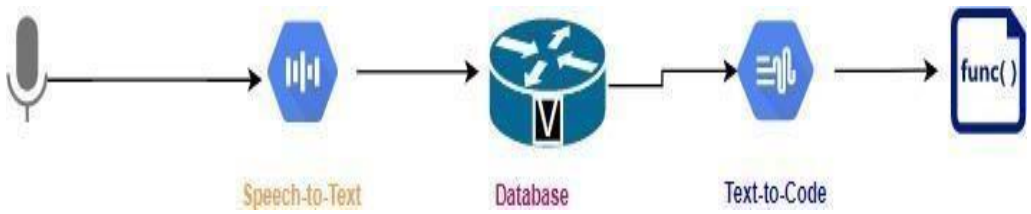


Figure 3.2: System Block Diagram

An overview of the system block diagram is shown in Figure 4.2. The related modules are displayed in the system structure, i.e

- Mic or any input device : To obtain the speech from the user
- Speech to Text Api : We use Webkit Api inorder to convert Speech to Text.
- Database : Here the obtained word is compared with the stored keyword in database.
- Text to Code Converter: Here the specific word if present in database then code is obtained.
- Coded output on IDE : Here the user's spoken code can be obtained on editor.

3.2.2 Data Flow Diagram

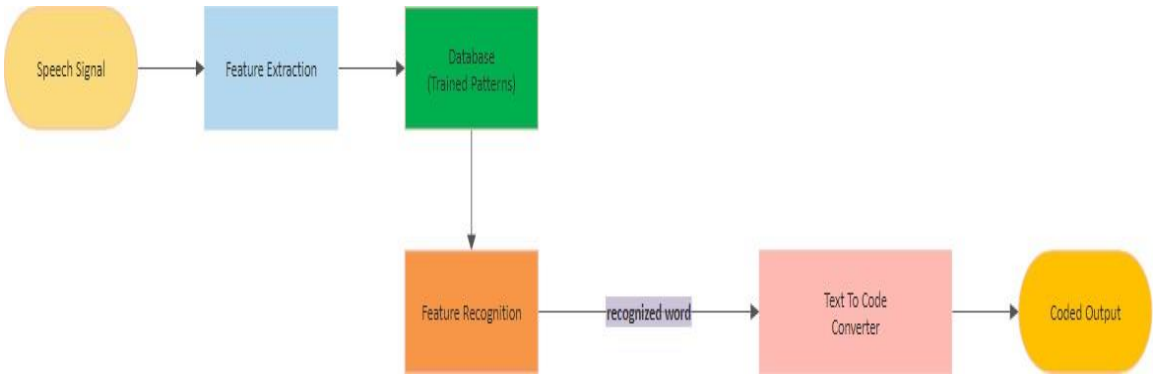


Figure 3.2.2 Data Flow Diagram

The direct data flow from the input stage to the final output stage is shown by the data flow diagram. Get a complete picture of your system implementation without having to go deep into complex tasks. Below is a step-by-step guide to your data. The data flow in the system is as follows:

- The voice will be recognized by the system .
- And that recognized words or words will be compared with the stored keywords in the database.
- If they are matched then that will be printed on editor.
- After this again by recognizing the specific keywords the program will be compiled and executed.

This system will be easy to use, it reduces human efforts and the use of hardware resources.We believe that IDE's with vocal programming support like ours can make programming accessible to anyone, regardless of physical ability.
All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102

IV. IMPLEMENTATION

4.1 Implementation Platform

4.1.1 Hardware Requirements

- **Processor:** Intel® Core ™ i3-8250U CPU @1.60GHZ 1.80GGHZ and Above
- **Installed Memory (RAM):** 8.00 GB and Above
- **Connectivity:** Wi-Fi (150Mbps and Above)

4.1.2 Software Requirements

- **Operating System:** Windows (10) (64bit)
- **Software Used:** Html , CSS , Bootstrap , Manoco Editor
- **Programming Languages:** JavaScript
- **Server:** Localhost Tomcat

4.2 Implementation Details

A high-level architecture design was created at the first design phase in order to combine the obtained functional requirements and non-functional needs for this project. Modeling has also been done for the system's fundamental workflow and architectural design patterns. The four parts of this system are as follows:
• Speech Recognition System to recognise speech commands from people
• A semantic analysis system to help build the software the individuals is requesting and help distinguish between catchphrases.

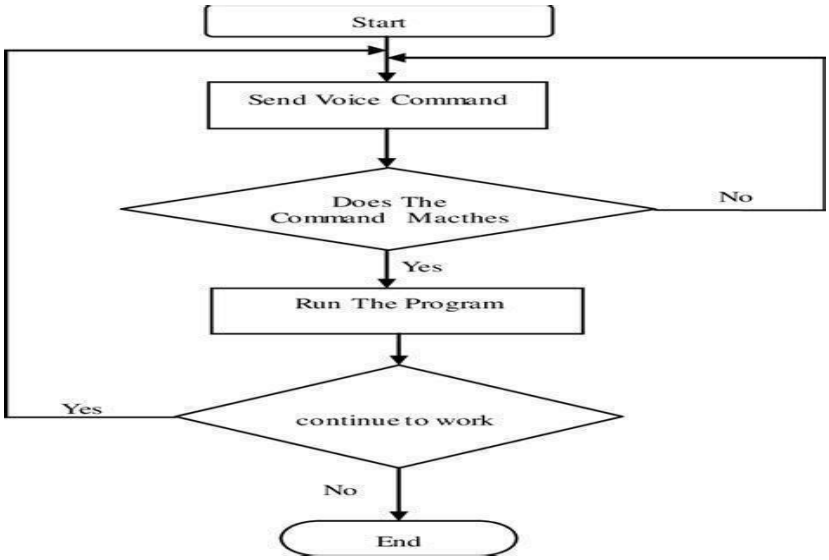


Figure 4.2 Implementation Structure

Speech Recognition System to recognise speech commands from people
In "Speech to Text interpretation," the typical scenario is to first set speech commands to the programming before catching the voice and muting the sound. According to the circumstance, the voice must first be recorded using a microphone, and the recorded voice must then be sent outside of the system. The speech sources of information need to be clearer in order to provide a more exact and audible evidence of sound signals. The captured voice should be put through a sound-reducing device in order to reduce the noise overall. The client's sound result is taken by subtracting the text from it using sound decrease. Discourse examination, as well as discourse and aggregation, are all covered by the Web Speech API. As a result, it supports both the transformation of speech into text and the reverse. The API is entirely written in JavaScript, one of the most popular client-side prearranging languages used today on the web. The Web Speech API is event-based, which blends in well with JavaScript's relatively callback-heavy coding style. The client specialist, who handles calls to the API, is also in charge of handling any communication with an online discourse acknowledgment service. Obviously, this assumes that the client specialist executes the API. The occasion-based design enables projects to process communication without doing so simultaneously. Additionally, occasions are used to deliver mid-discussion feedback

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022
DOI: 10.17148/IJARCCE.2022.116102

results, which is beneficial because it enables projects to provide the client with feedback almost immediately. Discourse acknowledgment can be prevented at any time, which is advantageous because it relieves the site designer of additional work in the scheduling of the event controller. The voice contribution from the client is converted to text using the Web Speech API. Web Speech Programming interface is only used in the "Vocal Programmer" project to recognise the client's English words. Any existing discourse to message framework that takes into account programming-related keywords cannot accurately capture these terms. If statement, integer main function, and other model-specific statements. The Vocal Programming framework prepares the programming-related words in an unusual way in order to capture client voice input accurately and effectively. In reality, when there is background noise in the system, it is difficult to hear the spoken input.

Semantic Analysis Engine to distinguish catchphrases and foster the construction of the program which client is inquiring

The principal usefulness of this module is to recognize the programming keywords and make the construction of the Programming structure. This should be possible by fostering a calculation to figure out all the conceivable programming word phrases, connected with programming language C. To foster a calculation like this, specialists ought to zero in on the super keywords and a method for recognizing the catchphrases. Additionally, need to zero in on the watchwords which are like one another. Assuming the words distinguished are comparable, utilizing the calculation needs to recognize these words and save it into a data set called Syntax database for additional handling. Agreeing to the given catchphrases coding design will be produced. In this segment, there are two principal parts to be thought of. Initial segment is Semantic Analysis and second part is Programming Idea module. Semantic Analysis is a hypothesis of information procurement enlistment and portrayal. It was first presented as a data recovery strategy .It is a programmed numerical learning strategy for dissecting the connections and likeness structures among reports and terms, depending on no human encounters, earlier hypothetical models, semantic word references, or information bases . This semantic examination module is a subsystem which will distinguish the keywords from the voice order that has been changed over from the above Speech to Text module. There are five fundamental functionalities in this module.

V. TESTING

In the testing phase, we tested the vocal programming application with many different commands and assessed the accuracy of the application. The testing was done on all the functionalities of the application, which are:

5.1 Testing on Libraries:

During this phase, the libraries were tested. In order to include libraires in the application, the user has to say the word include and the name of the library only without saying the words like hash or ".h". But when the user says these words, the application detects these words but it does not print any because it is not required. When we say these words, nothing will be printed on the editor, hence following the readme manual will help the user in using the vocal programming application without any problem. During the testing, we tested for a total of 20 times by saying the names of some libraires and the accuracy was around 90%. Most of the time, the application was able to recognize the words and sometimes it was recognizing something else due to noise or other factors.

5.2 Testing on Variables:

During the testing of the data types, we tested the application with different data types along with the different names for the variable names. In order to print the datatypes, we have to say the word integer or int if you have to create a variable of the type integer, character for char and so on. Later to create a variable say the name of the variable and to assign a value to it say the name of the variable and say equals the value of the variable. During the testing, we found out that sometimes when we say equals the word equals itself was getting printed instead of the equal sign, so we included a dictionary in the speech recognition file which included many words which sound similar and included equals also there. Later the accuracy of the variables was recorded to be around 95%.

5.3 Testing on Functions:

During the testing of the functions, a series of functions were tested including the main function. As mentioned earlier the computer recognizes some words as similar for example, when we say function integer main, the application should print int main() but sometimes due to noise instead of main, name or mean was getting added. To overcome this problem also, we included the words similar to main in the dictionary. After that, we were able to get accurate outputs. The accuracy was around 85%.

5.4 Testing on Printf:

During the testing of the printf statement, we tested the printf statement with many messages and even printing the values of the variables. To use the printf function, start the command with the word print. To use a variable in the quote, use the

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102

word percent/percentage in the quote. No need to use the identifier, such as 'd'; identifiers will be added automatically based on the variable's data type. To specify the variable to be used instead of the placeholder, use the word variable. Following the word variable, say the name of the variable to be used. To add a newline (\n) in the print quote, use the word line. In the testing, some of the print statements were displaying something else that was not spoken and sometimes the variable's value was not printing. The accuracy was around 80% for the printf statement.

5.5 Testing in Scanf:

During the testing of the scanf statement, a series of variables were given as input using voice commands. In order to use the scanf statements, the user has to say the following: Follow the word with the names of variables to be scanned. Format identifiers or a quote for scanf is not required. The quote will be written automatically based on the data types of variables. The word ampersand/and is not required before each variable either. The '&' symbol is added automatically before each variable. As mentioned in the printf, even the scanf statement was giving some errors and the variable was not getting recognized by the system. The overall sccuracy of the scanf statement was around 85%.

5.6 Testing of If-Else:

In this phase the working of the if-else statements were monitored and the input was given on some different if else statements. In order to use the if else statements, the user has to give the following commands: start the command with the word if or else. Follow the word with the condition. Proceed variable names with the word variable, and say constants (integers or chars) directly. To add relational operators, use the words equals, greater, greater equals, less or less equals. The overall accuracy of the if else statement was around 90%.

5.7 Testing of While Loop:

During the testing of the while loop, different conditions were given as voice commands to the voice programming application. To include the while loop in the application, the user has to fol

VI. EXPERIMENTATION AND RESULTS

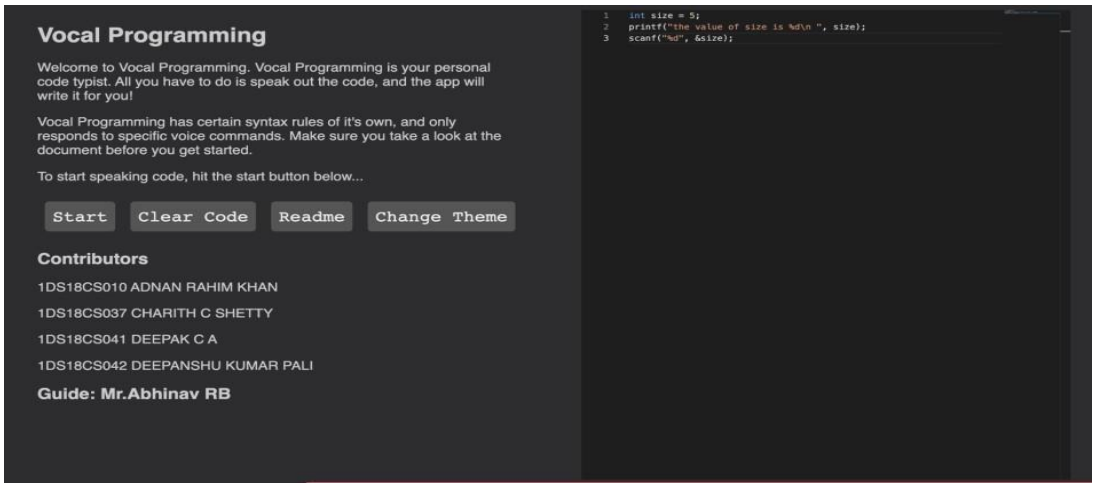
6.1 Testing on various vocal commands

- Test Case 1:

Here we tested the application with writing atomic commands like declaring the variables, printing commands and reading commands.

Voice command: integer size equals 5
Statement Written: int size = 5;
Voice command: print the value of size is percent line variable test
Statement written: printf("the value of size is %d\n ", size);
Voice command: scanf size

Statement Written: scanf("%d", &size);



- Test Case 2:

Here we tested the application with a writing library including commands and declaring functions.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

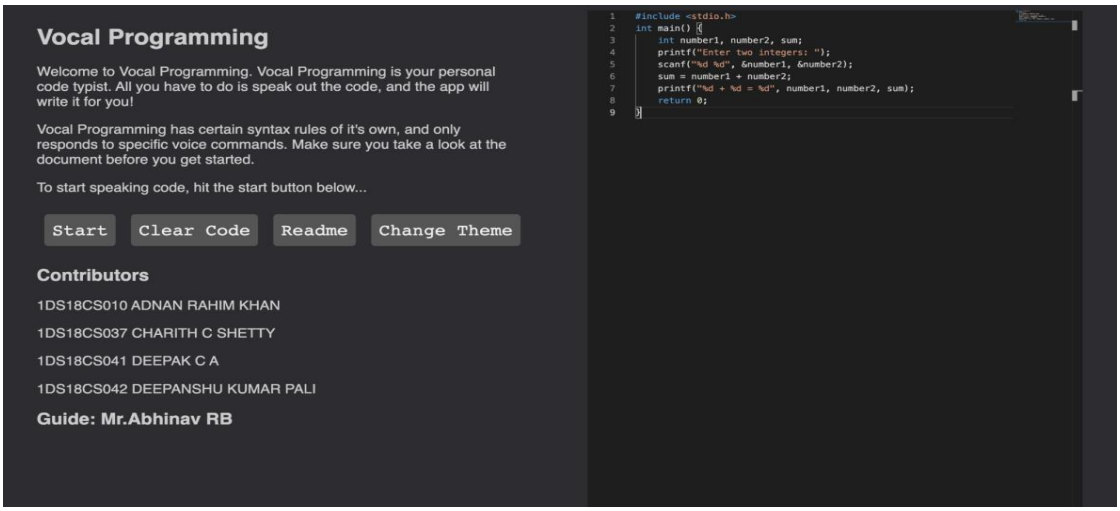
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102

Voice command: include stdio
Statement Written: #include<stdio.h>
Voice command: function integer main
Statement written: int main() {}

- Test Case 3:

Here we tested the application by writing a complete working C program to add two numbers.



VII. CONCLUSION

Introducing an IDE that supports language programming and showing that you can use the system for programming. We contribute to the study of empirically designed language programming systems and their ease of use and potential for people who have physical limitations in using their fingers for the typing. Address user issues while developing computer programs. Developing computer programs is not an easy task and requires hardware resources that the user must handle. The user's finger may be injured while entering the code continuously. To avoid this problem, design a tool that can develop computer programs by voice. The voice is recognized by the system, the recognized words are compared to the keywords stored in the database, printed in Ide if they match, then the program is compiled, recognizes the specific keyword and runs again. increase. This system is easy to use and reduces the use of personnel and hardware resources. We believe that with the support of language programming like us, the IDE will be accessible to anyone, regardless of their physical fitness.

Future Enhancements:

- Improving the Software to be more Accurate.
- Implementing for various Programming languages.
- Addition of Compiler inorder to compile the program.
- File Traversal System by Voice
- More Voice commands inorder to ease the use of software

ACKNOWLEDGMENT

We would like to sincerely thank our guide : **Professor Abhinav R B** and our committee members: **Dr. Vindya Malagi**, Professor **Ramya R S** for their guidance through this process

REFERENCES

[1] Arnold, S. C., Mark, L., & Goldthwaite, J. (2000, November). Programming by voice, VocalProgramming. In Proceedings of the fourth international ACM conference on Assistive technologies (pp. 149-155). ACM.
[2] Begel, A. and Graham, S. L., (2005) Spoken programs,In Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05), 2005, pp. 99- 106.
[3] Begel, A., & Graham, S. L. (2006). An assessment of a speech-based programming environment. In Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on (pp. 116-120). IEEE.

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.116102

[4] Brandt, J., Dontcheva, M., Weskamp, M., and Klemmer, S.R. Example-centric programming: integrating web search into the development environment. ACM, New York, New York, USA, 2010.

[5] Brandt, J., Guo, P.J., Lewenstein, J., Dontcheva, M., and Klemmer, S.R. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. ACM (2009), 1589–1598.

[6] Brault, M.W. July 2012. Americans With Disabilities: 2010. U.S. Department of Commerce.

[7] Curatelli, F. and Martinengo, C. (2010) Enhancing digital inclusion with an 9nglish pseudosyllabic keyboard. HCI in Work and Learning, (2010).

[8] Dai, L., Goldman, R., Sears, A., and Lozier, J. Speech-based Cursor Control: A Study of Grid-based Solutions. SIGACCESS Access. Comput, 77-78 (2003), 94–101.

[9] Désilets, A. (2001). VoiceGrip: a tool for programming-byvoice. International Journal of Speech Technology, 4(2), 103- 116.

[10] Désilets, A., Fox, D. C., & Norton, S. (2006, April). VoiceCode: an innovative speech interface for programmingby-voice. In CHI'06 Extended Abstracts on Human Factors in Computing Systems (pp. 239-242). ACM.

[11] Feng, J., & Sears, A. (2004). Using confidence scores to improve hands-free speech based navigation in continuous dictation systems. ACM Transactions on Computer-Human Interaction (TOCHI), 11(4), 329-356.

[12] Findlater, L., Moffatt, K., Froehlich, J.E., Malu, M., and Zhang, J. Comparing Touchscreen and Mouse Input Performance by People With and Without Upper Body Motor Impairments. ACM Press (2017), 6056–6061.

[13] Gorter, J.W., Rosenbaum, P.L., Hanna, S.E., et al. Limb distribution, motor impairment, and functional classification of cerebral palsy. Developmental Medicine and Child Neurology 46, 7 (2004), 461–467.

[14] Harada, S., Landay, J.A., Malkin, J., Li, X., and Bilmes, J.A. The Vocal Joystick:: Evaluation of Voice-based Cursor Control Techniques. ACM (2006), 197–204.

[15] Harada, S., Wobbrock, J.O., and Landay, J.A. Voicedraw: A Hands-free Voice-driven Drawing Application for People with Motor Impairments. ACM (2007), 27–34.

[16] Holmes, R. and Walker, R.J. Systematizing pragmatic software reuse. ACM Transactions on Software Engineering and …, (2012).

[17] IGDA. October 2005. Game Developer Demographics: Exploration of Diversity. U.S. Department of Commerce.

[18] Jackson, J., Cobb, M., and Carver, C. Identifying top Java errors for novice programmers. Frontiers in Education, (2005).

[19] Johansson, V. 2008. Lexical diversity and lexical density in speech and writing. Working Papers 53. 61-79 pages.

[20] Kim, M., Bergman, L., and Lau, T. An ethnographic study of copy and paste programming practices in OOPL. … Engineering, (2004), 83–92.

[21] LaToza, T.D., Venolia, G., and DeLine, R. Maintaining mental models: a study of developer work habits. (2006).

[22] Leopold, J. L., & Ambler, A. L. (1997, September). Keyboardless visual programming using voice, handwriting, and gesture. In Visual Languages, 1997. Proceedings. 1997 IEEE Symposium on (pp. 28-35). IEEE.

[23] MacKay, D. Dasher–an efficient keyboard alternative. Advances in Clinical Neuroscience and Rehabilitation, (2003).

[24] MacKenzie, I.S. and Zhang, S.X. The design and evaluation of a high-performance soft keyboard. ACM, New York, New York, USA, 1999.

[25] Mathiowetz, V., Volland, G., Kashman, N., & Weber, K. (1985). Adult norms for the Box and Block Test of manual dexterity. American Journal of Occupational Therapy, 39(6), 386-391.

[26] Maulsby, D., Greenberg, S., & Mander, R. (1993, May). Prototyping an intelligent agent through Wizard of Oz. In Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems (pp. 277-284). ACM.

[27] Nicolau, H., Guerreiro, J., and Guerreiro, T. Stressing the Boundaries of Mobile Accessibility. 2014.

[28] Ruan, S.,, Wobbrock, J.O., Liou, K., Ng, A., and Landay. J.A. (2016) Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices. arXiv:1608.07323 {cs.HC} (Aug. 2016).

[29] Sears, A., Feng, J., Oseitutu, K., & Karat, C. M. (2003). Hands-free, speech-based navigation during dictation: difficulties, consequences, and solutions. Human-computer interaction, 18(3), 229-257.

[30] Sears, A., Revis, D., Swatski, J., and Crittenden, R. Investigating touchscreen typing: the effect of keyboard size on typing speed. Behaviour & … 12, 1 (1993), 17–22.

