International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

SpeakEasy – Vocal Coding Online Platform

Siddhesh Borkar[1], Rhytham Kabra[2], Smeet Kevadiya[3]

[1,2,3]*Dept. of I.T. Engineering, Xavier Institute of Engineering, Mumbai University, India*

Abstract - *Programming by voice with current Speech Recognition (SR) systems is awkward because programming languages aren't meant to be spoken. Here, we describe various usability problems with programming-by-voice and show that none of the present programming by voice tools addresses all of those barriers. We then present Speakeasy, a programming-by-voice tool that addresses the widest range of programming-by-voice problems to this point. Speakeasy uses a singular approach where programmers' first dictate code using a straight forward pseudo-syntax, and then translate that automatically to native code within the appropriate artificial language. Finally, even though Speakeasy is the tool that currently addresses the widest range of programming by voice problems, we conclude that a stronger tool are often developed by combining features of Speakeasy with features of other existing programming by voice tools.*

1. INTRODUCTION

SpeakEasy is a web-based voice-based programming platform which enables users to use their voice to code. The inspiration for this application comes as a preventive solution to reduce Repetitive Strain Injury. Activities like continuous typing can lead to injuries in the wrist and finger muscles called Repetitive Strain Injury, a broad term for a vast number of injuries. These injuries start with symptoms like wrist pain and in severe cases can lead to loss of sensation in the wrist and fingers also called Carpal Tunnel Syndrome. SpeakEasy provides the user a preventive tool and also reduces vocal stress on the user with the use of predefined commands.

Adding to SpeakEasy's capabilities can also be beneficial for specially abled people who might find it difficult to use the traditional keyboard-mouse technique of operating. SpeakEasy uses technologies like Node.js, Javascript and its packages for its backend, while its frontend is designed using HTML and CSS.

2. PROBLEM STATEMENT

In recent years, there has been an increase within the amount of computer programmers affected by from Repetitive Strain Injury (RSI) - an umbrella term covering a series of musculoskeletal disorders caused by repetitive motion of the hand, and arms. In today's world, we need to get a specific errand completed as soon as possible, likewise for software engineers or any individual from a technical background wants to do their errands with less effort and higher pace. For those individuals, or any programmer with a handicap that precludes keyboard and/or mouse input, we can provide a chance for them to code, to improve the productivity of coding, to diminish the frenzied writing efforts we can implement Speech Recognition (SR), which is an attractive alternative because it could allow them to try and do their work without using such devices and perform nearly all errands that you can do ordinarily.

3. REVIEW OF LITERATURE

In the past, most research is focused on finding the methods, problems that arise for the implementation and the proposed solutions for the problems.

One work focuses on the development of a system which acts as an aid for programmers suffering through RSI(repetitive strain injury) caused due to repetitive typing, the proposed system not only provides solution to usability problems which include punctuations, existing symbols, local navigation, new symbols but also provides portability solutions with cross SR and cross editor.

Another paper proposes a design that generates environments that enables people to program by voice and a method of determining if the system is successful. It also shows how these generators are often used to support entering data and writing XML documents.

"VoiceGrip: A Tool for Programming-by-Voice", this paper describes various usability problems with programming-with-voice. It then describes VoiceGrip, a programming-by-voice tool that addresses the widest range of programming-by-voice problems to date. The proposed system in paper is able to solve all the problems except two. The system has two commands

namely Compilation and translation commands, the compilation commands are used to scan a series of source files for programming symbols, by forwarding to the Source analyser which scans the source files, identifies all the symbols they contain and stores them as Symbol Database.

Stephen C. Arnold & Leo Mark and John Goldthwaite "Programming by Voice, Vocal Programming", The proposed design consists of a generator for voice recognition syntax-directed programming environments. The generator takes an input context-free grammar (CFG) for a programming language, such as Basic, C, C++, Java or XML DTD together with a voice vocabulary for that language. The second domain the paper focuses on is the use of speech recognition for data entry. It can successfully enter data in an XML document which has been demonstrated in the paper. A number of various sort of commands are needed within the syntax-directed programming interface also as within the forms-based data entry and editing interface.

M. A. Jawale, A. B. Pawar, D. N. Kyatanavar "Smart Python Coding through Voice Recognition", the proposed system in paper, will support the specific operation that is developing a computer program through the voice of the user, which is recognized by the system. Keywords are going to be identified as Key in voice input and can be mapped to a prepared dictionary database of stored keywords. If it is found in this set, it will be displayed in the user workspace for further processing otherwise an error message will be given to the user with prompting for correct voice input.

Anurag Singh, Ganesh Tambatkar, Shashank Hanwante, Nitish Agrawal, Rahul Hajare, Ketki Khante "Voice to Code Editor Using Speech Recognition", in the proposed paper PyAudio is a Python by default speech recognizer engine which is extremely helpful and may recognize a phrase or sentence easily and that we also can improve its productivity by defining macros in order that its understandability should get improve. In this paper, they have categorised the system into Front End Analysis and Back End Analysis for improved understanding and representation of speech recognition systems in each part. The main reason for them to classify the system is that it'll provide higher accuracy because they were facing issues associated with noise, vocabulary size, and domain in order that they came up with their idea of Front-End Analysis and Back End Analysis.

5. PROPOSED SYSTEM

Programming environment can create annoying complications for the software developers who suffer from repetitive strain injuries (RSI) and related disabilities which makes typing difficult. The software development process is very much repetitive and has activities like program composition, editing, and navigation, and the tools used for programming are solely based on keyboard typing and offline (i.e desktop applications). This would mean a lot of physical interaction with the keyboard and mouse and also the tools have to be installed on the developer's machine which means all the processing will be completely performed on the user machine.

The main purpose of this project is to reduce the efforts in developing software and programs by reducing the physical problems or restrictions associated with it and also to make it available to more and more people even if the user doesn't want to install it on their own machine instead leverage the online servers to process their code. Our system is based on Client-Server architecture as most of the modern web apps nowadays. Here the processing is divided between client and the server so as to take the load off from the client side.

5.1 CLIENT SIDE

Client side acts as a regular modern GUI that you come across any websites present online which provides an easy to use visual interface for users. The GUI is lightweight so as to keep the load time as low as possible.

The client side consists of a Code Editor (CE) where user can code using their voice and also with physical interaction devices if they want. Client side includes a Voice Recognition Module (VRM) as well which turns on if the appropriate permissions are provided from the browser. This VRM will detect voice continuously and only stop detecting intelligently after the user has stopped speaking. VRM also has Speech-To-Text abilities which will interpret the speech and convert it into text. This interpreted text is then sent to the server side for processing. This is the major part of the client side. When the processing is completed on server side and the response has been sent back to client, it is fed

into a Command and Code Separator (CCS) module which recognizes the commands or operations to perform on client side editor and what data is to be displayed in the code editor. The user can send the code to the server side for execution and respective output will be received back to the user.

5.2 Communication between Client-Server

As this system is based on client-server architecture, there has to be a communication link between them. We have used sockets as our real time communication link. Each user is given a unique isolated socket (link) connection to the server, which ensures safety of users to some extent. This link is bidirectional so that both client side and server side can communicate with each other over the same link. When the STT has generated the text, it is sent to the server using this socket. Also the generated code and output is sent back to the client via the same socket.

5.3 Server Side

This is the heart of the system where most of the processing is handled. As soon as the STT interpreted text is sent to the server via socket, the server generates the code using the Code Generation Module (CGM). The CGM analyses and parses the text to look for relevant information on the basis of which the code will be generated dynamically. The CGM makes use of a command storage object where various code and its syntaxes are mapped, it is modified according to need and the generated code is then sent back to the user via the same socket the request came through. The other part of the server side handles the code execution, where the received code is stored on the server temporarily and then executed in the respective execution environment. The output of which is sent back to the user as soon as it is generated.

6. FUTURE SCOPE

Improving the accuracy of the microphone and reducing background noise to increase the ability to detect correct words.

Extending SpeakEasy's range from a Python only platform to add more languages.

AI powered autocomplete to increase the usability of the platform and further decrease vocal load of the user.

Integration with existing IDE's to leverage the existing processing power of IDE and introduce a vocal solution.

SpeakEasy's capability can be used beyond programming to facilitate a solution to reduce typing strain for people who aren't involved in coding.

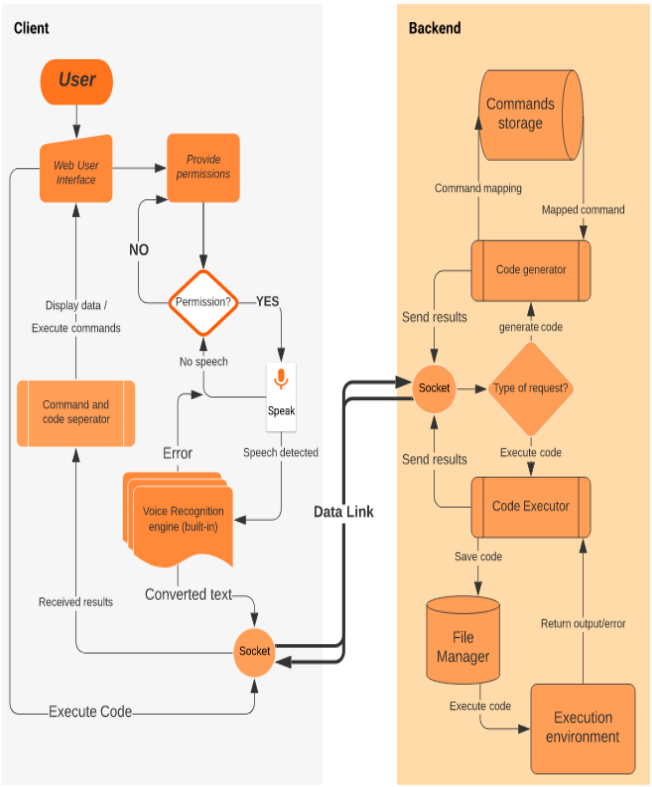

Fig -1: Process Flow of Speakeasy

7. CONCLUSION

As the world moves towards voice-enabled devices including smart home-devices and mobile assistants, SpeakEasy provides an unorthodox platform for coding in Python, the system allows users to either choose from a predefined set of commands or allows the user to speak the entire syntax himself. SpeakEasy reduces the processing load on the client by processing the code at the backend. The current implementation of SpeakEasy provides support for python only and is aimed to be extended for other languages for better user experience. The system doesn't possess any intelligence in the current version and can be introduced to enhance user operability.

REFERENCES

[1] A. Desilets "VoiceGrip: A Tool for Programming-by-Voice", in International Journal of Speech Technology, 2001.

[2] Stephen C. Arnold & Leo Mark and John Goldthwaite "Programming by Voice, Vocal Programming", Conference Paper, 2000.

[3] M. A. Jawale, A. B. Pawar, D. N. Kyatanavar "Smart Python Coding through Voice Recognition" in International Journal of Innovative Technology and Exploring Engineering,2019

[4] Anurag Singh, Ganesh Tambatkar, Shashank Hanwante, Nitish Agrawal, Rahul Hajare, Ketki Khante "Voice to Code Editor Using Speech Recognition" in International Research Journal of Engineering and Technology, 2018

[5] Rinor S.Maloku, Besart Xh.Pllana "HyperCode: Voice aided programming" in IFAC-PapersOnLine, 2016

[6] Liu Qigang, Xiangyang Sun "Research of Web Real-Time Communication Based on Web Socket" in International Journal of Communications, Network and System Sciences, 2012

[7] Nitin Washani, Sandeep Sharma, "Speech Recognition System: A Review", International Journal of Computer Applications, 2015.

[8] Désilets, A., Fox, D. C., & Norton, S. "VoiceCode: an innovative speech interface for programming by-voice", 2006

[9] MDN Web Docs - "the Web Speech Api", accessed 06 January 2021, <https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API>

[10] Node.js Docs, accessed 07 January 2021, <https://nodejs.org/api/child_process.html>

[11] SOCKET.IO Docs, accessed 09 January 2021, <https://socket.io/docs/v4>

[12] Express Docs, accessed 09 January 2021, <https://expressjs.com/en/4x/api.html>

[13] Ace Docs, accessed 27 January 2021, <https://ace.c9.io/#nav=api>