Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 269 (2025) 1211–1219

Procedia Computer Science

www.elsevier.com/locate/procedia

The 10th International Conference on Computer Science and Computational Intelligence 2025

CodeWithDisability: A Voice Command-Based Integrated Development Environment for Hands-Free Programming

Param Shukla[a,*], Krish Shah[a], Milan Haria[a], Mitansh Kanani[a], Ruhina Karani[a]

[a]*Department of Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India*

Abstract

Programming is heavily dependent on conventional input devices, which act as a hindrance for physically disabled individuals. Currently available tools such as screen readers and voice typing provide minimal coding assistance because programming tasks are very complex. CodeWithDisability, a voice-command IDE proposed in this study, allows users to code, edit, compile, and run code completely via speech. Developed with Electron.js and Python, the system translates verbal commands into speech recognition and converts them into C syntax through a well-structured dataset. The current implementation includes a dataset that is exclusively for C programming. Given that the system has a flexible architecture, it is able to integrate other programming languages at a later date.

Keywords: Voice-Based Coding, hands-free coding, voice-command programming, disability-friendly coding platform, accessibility in programming, speech recognition.

* Corresponding author. Tel.: +91-982-038-5988 .
E-mail address: shuklaparam09022004@gmail.com

10.1016/j.procs.2025.09.062

1. Introduction

Learning to program is so critical in the ever-increasing digital world we live in; it is what creates programs that are so omnipresent and govern all day-to-day operations [1]. But conventional programming environments are motivated nearly exclusively by mouse and keyboard input and are inaccessible to individuals lacking hands [2], [3]. A major obstacle that most physically disabled individuals encounter in participating in software development there are no specialized tools to code with your hands-free [4]. While general purpose assistive technologies like screen readers and speech-to-text have assisted in minimizing barriers to everyday computing, these are insufficient for programming [3], [5], [6]. Programming requires careful syntax and indentation, along with arcane characters that speech-to-text tools are not well-suited to deal with [2], [7]. Consequently, people with physical disabilities continue to be underrepresented in the tech community, both in terms of opportunities and software development contributions [4].

To solve this issue, this research designs CodeWithDisability, a voice-controlled Integrated Development Environment (IDE) for hands-free coding. As opposed to the common voice-to-text software, which can assist with textual input, but not with programming, the system is engineered to enhance coding efficiency, allowing you to utilize natural voice commands to generate and modify code [2], [5], [8]. The mechanism of the system is directed by a language model that establishes various slots of utterances conveying pre-defined voice instructions to programming syntax constructs [5]. It corrects a problem with the majority of speech recognition tools, which frequently misidentify commands unique to programming [7], [9]. CodeWithDisability IDE has been developed using Electron.js for the GUI and Python for speech processing, which makes it a lean and efficient tool for hands-free coding. The IDE currently supports C programming, enabling users to write, edit, and execute C programs with no physical input devices [5], [10]. Speech recognition is used to convert voice commands into formatted C code so that programmers with disabilities are able to type code without employing a keyboard or mouse [2],[11]. File operations such as creation, saving, and execution of source files are also performed entirely through voice commands, ruling out the need for manual intervention [5],[12].

The lack of inclusive programming tools has been a significant hindrance to inclusion for individuals with physical disabilities, keeping them away from participation in the tech sector [3], [4]. Most available accessibility solutions provide some form of basic assistance but mostly need manual tweaks, or extra hardware (largely impractical with no hands) to function fully making them impractical [4], [6]. CodeWithDisability bridges this gap through a fully voice-enabled programming experience and allows people with disabilities to engage in the software development process in the same manner as non-disabled individuals. This system eliminates physical barriers to coding, thus enhancing inclusion in technology for people with disabilities and enabling opportunities for programmers [1]. This work points to the possibility of hands-free coding environments to enhance software development accessibility [2], [4]. The system discussed focuses on the fact that speech-driven programming can serve as an alternative to conventional input modes of programming and breaks the door open to future improvements in assistive technology [2], [7], [11]. Through its provision of a specialized platform for voice coding, CodeWithDisability is geared with a more diverse and inclusive technology environment for individuals in society with disabilities who want to be self-actualized as software engineers [1], [4].

2. Related Work

This author in [5] describes presents a web-based "Vocal Programming" system that allows users, especially those with physical limitations, to write C programs via voice commands. Using the Web Speech API, the system converts spoken language into executable C code with 85–95% accuracy in quiet settings. It supports common C constructs like functions and conditionals. Future work includes expanding to other languages, enhancing voice recognition, enabling voice-controlled file operations, and customizable UI for better accessibility.

The author in [7] study assesses the educational impact of voice programming using ASR with 96 undergraduates. Voice input reduced negative attitudes toward programming but lowered self-efficacy. It improved coding speed and accuracy on simple tasks but underperformed compared to keyboard input on complex ones. The study suggests voice programming can support early learners but requires more research, especially for accessibility, hybrid input methods, and long-term effects. Gertsman and Holzinger write in [4] on integration of principles of accessibility in modern software design practices, highlighting both opportunities and challenges problems that arise in this relationship. The study examines the growing demand for inclusive software systems and how accessibility

problems are on the periphery of agile and fast-paced development environments. Their work, presented in Lecture Notes in Computer Science (Vol. 12765), critically analyzes the limitations of the existing tools, the absence of standardized processes, and the insufficient training of developers in accessibility principles. They believe that effective integration involves not only technical means but also cultural and changes in education within development teams. The authors spot future opportunities through automation,frameworks, and more stakeholder involvement, generating a sound argument in favor of a more systematic and coherent approach to software engineering.

The study detailed in [9] interviewed the diversity voice coding, specifically how the novices and experts use code, without precise grammatical requirements. The experiment tried two of the data collection protocols: saying the highlighted lines of code and stating the lines of code that were skipped. The results show that programming experts utilize code in a more natural language way while beginners look more tightly connected to syntax. Education of a commercial speech recognizer as language model, on transcripts of spoken code helped reduce the error rate by 27% from baseline. The research showed the ambiguity of natural language as in the case of oral code, with the necessity of firm models of language to improve accuracy for voice programming systems.

3. Methodology

CodeWithDisability is developed in a systematic manner strategy to offer an Integrated Development Environment (IDE) with voice-controlled and hands-free mechanisms for people with motor disabilities. This section deals with the system architecture, implementation, the speech recognition process, command dataset, and execution pipeline for smooth code dictation, editing, and execution.

3.1. System Architecture

The system is designed in a modular way to provide a programming experience that is hands-free and voice-driven as shown in Fig 1. The User Interface (UI) is built in a browser utilizing React and Electron.js, allowing for an intuitive coding environment. Once the user issues a command, they can see their code reflected in real time. The Speech Recognition module is implemented in Python utilizing the speech_recognition library. The Speech Recognition module waits for the user to issue a command and captures the voices and turnsit into text. The Command Processing Unit interprets the command and following the established rules, the recognized speech is tied to certain commands written in rules in JSON files. The rules convert spoken commands into syntactically correct C code or commands to the IDE. The Code Execution Engine utilizes GCC (GNU Compiler Collection) to compile and execute the code. The File Management System is built-in allowing. The user to create, save, open or edit a code file using a voice command and will assist with file management by creating a directory to aid in a structured way of managing files.

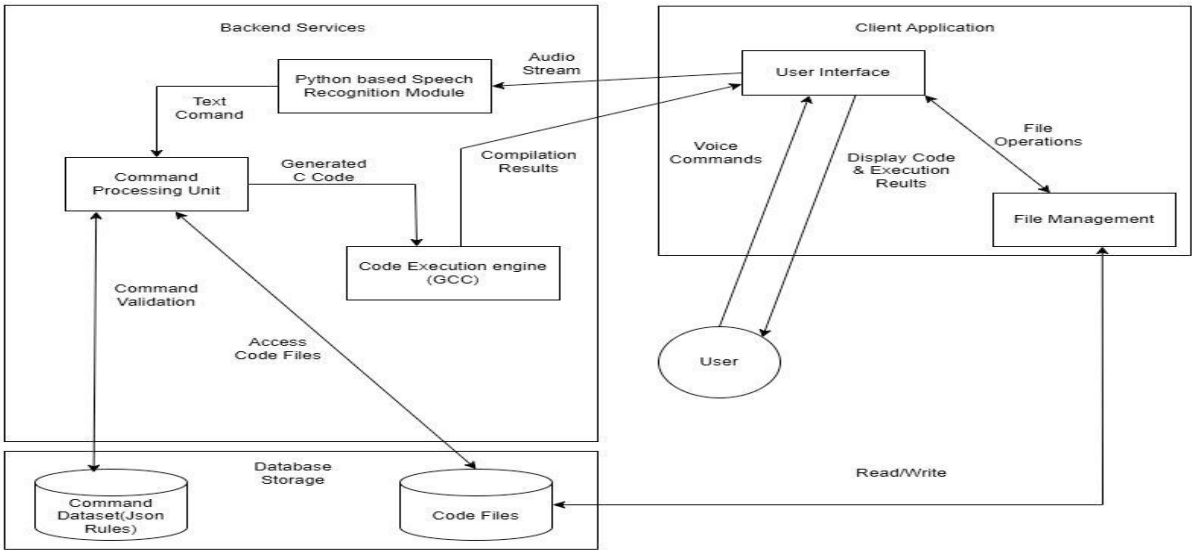

Fig 1. System Architecture

3.2. Voice Command Processing

The system is always listening for voice commands through the Speech Recognition Module. Once a command is accepted, the Speech Recognition Module interprets the command to decide what the user would have wanted to do, either open a new file, modify a current file, or run some code. The Google Web Speech API translates spoken words into text, which is the initial step towards further processing. This step ensures that the system can only observe user's commands and prepare them for the subsequent stage of the process.

3.3. Speech-to-text Conversion & Command Mapping

After the voice input is transcribed into texts, it will be mapped to a dataset.json file. The dataset only contains valid syntax for programming constructs and can convert spoken words into executables. For example, the spoken command, "declare integer x", is mapped into valid C syntax: int x;. If the dataset contains an entry matching the spoken command, the system will create the code snippet, and insert it into the active file. The mapping between the spoken command, and the dataset, allows the system to account for a very large range of coding commands accurately and easily.

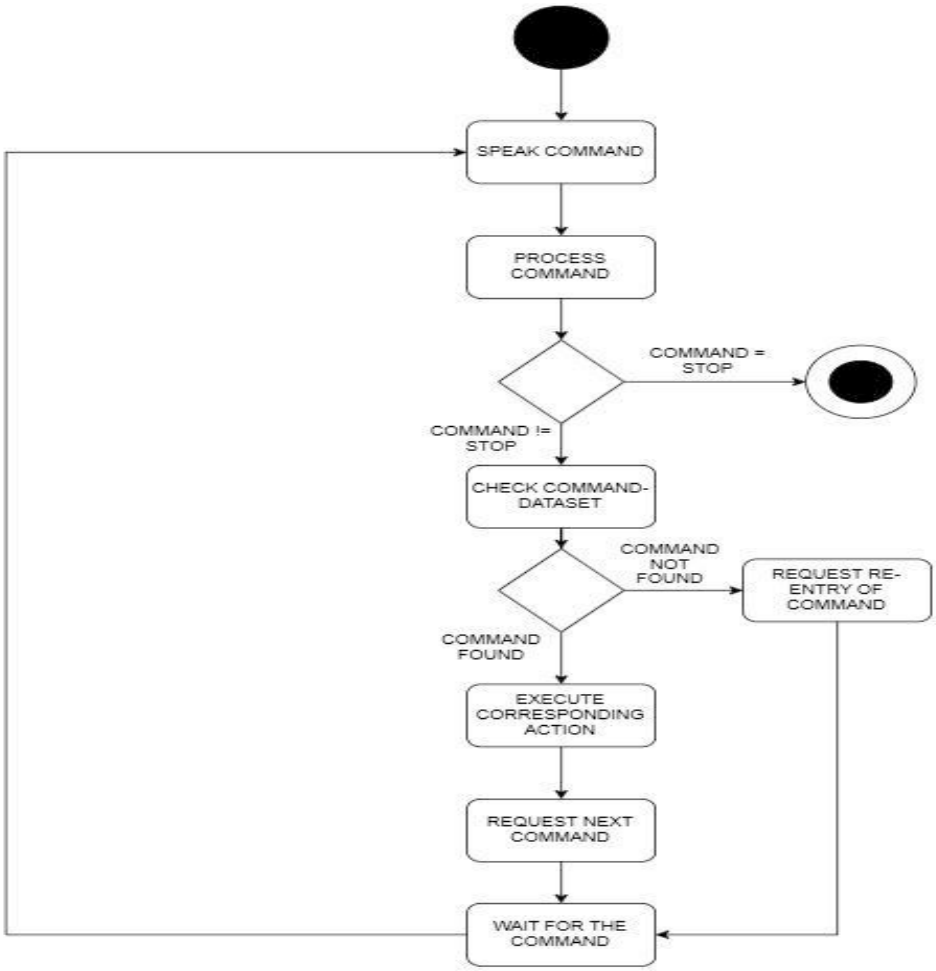

Fig 2. UML Activity Diagram

3.4. Code Execution & Debugging

When commands are put into code, users compile and run their programs using the Code Execution Engine. The system compiles and runs with GCC and gives real time output or error feedback data. When compilation errors exist are used within the code, the system adequately warns the user, offering little debugging capability. For example, if the user says "Execute" the system compiles the current file and then employs it, and shows any results or issues that come from its Execution enables users to engage in a seamless process which enables movement toward from code dictation to execution.

3.5. Code Execution & Debugging

File Management System allows the user to control their code files completely using voice commands. Users are able to make new files, open existing files, and save their work. The system offers efficient file organization by an organized directory. For example, whenever a user gives a command to create a file, and it generates a new file from a default template, allowing the user to start programming immediately. Similarly, users can read files in order to show or change their contents, the system managing file existence checks to prevent errors. This functionality makes users capable of managing their projects effectively without requiring manual input.

3.6. Dataset for Voice Commands

The system relies on two JSON data sets, command_mapping and actions_mapping, that translate speech commands to programming constructs to IDE actions respectively. These figures are utilized to ensure a voice command will be identified and subsequently translated into valid programming syntax or proper IDE action. Below is an example of the dataset structure.

Table 1. Example from Command_mapping dataset

Spoken Command	Mapped Code Snippet
declare int	int x;
print string name	printf(“%s”,name);
get integer input for age	printf(“Enter value for age:”); scanf(“%d”,&age);
if statement with x > 0	if(x>0) { }
include library stdio	#include<stdio.h>

Table 2. Example from actions_mapping dataset

Spoken Command	MappedAction
start coding	Activates coding mode
execute	Executes code file
save file	Saves the current file
create new file	Creates new file

The dataset also includes alternative phrases to accommodate variations in user speech. For example, both "declare integer x" and "initialize integer x" map to the same C syntax. Additionally, validation rules are incorporated to prevent incorrect or incomplete command execution. If an ambiguous or partial command is detected, the system prompts the user for clarification, ensuring code integrity and reducing syntax errors. Designed with scalability in mind, the dataset can be extended to support additional programming languages and IDE functionalities, making the system adaptable for broader use cases in voice-based coding.

4. Results & Discussion

4.1. System Implementation & Demonstration

The practical implementation of CodeWithDisability clearly shows the successful use of voice commands for multiple programming tasks. As shown in Fig 3, the "Create New File" dialog box indicates to the user that they have issued a voice command (to create a new file). This interface allows the users to specify the file name and programming language type without the need for any use of the keyboard.

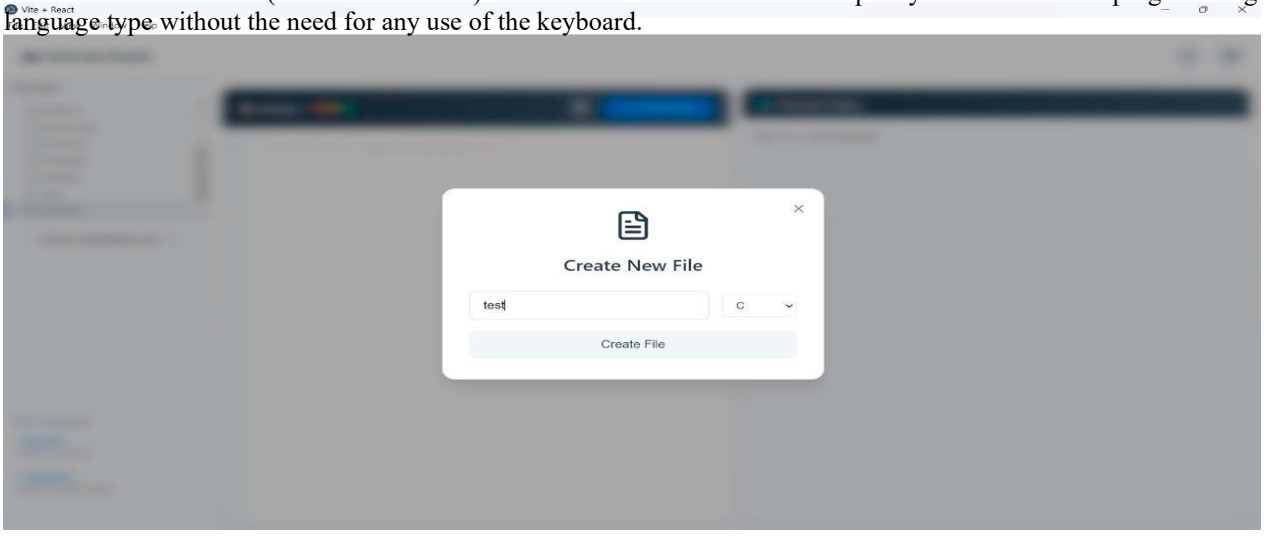

Fig 3. Create New File dialog activated by voice command

Fig 4 displays the IDE with a successfully written C program that checks if a number is a palindrome. This code was entirely generated through voice commands, with the appropriate command dictated by the user and accurately converted to syntactically correct C code. The terminal output on the right confirms successful compilation and execution of the program, showing the result "121 is a palindrome." The execution itself was also triggered through voice commands when the user spoke "Run Code". The system responded by compiling and running the program without requiring any physical interaction with the computer.

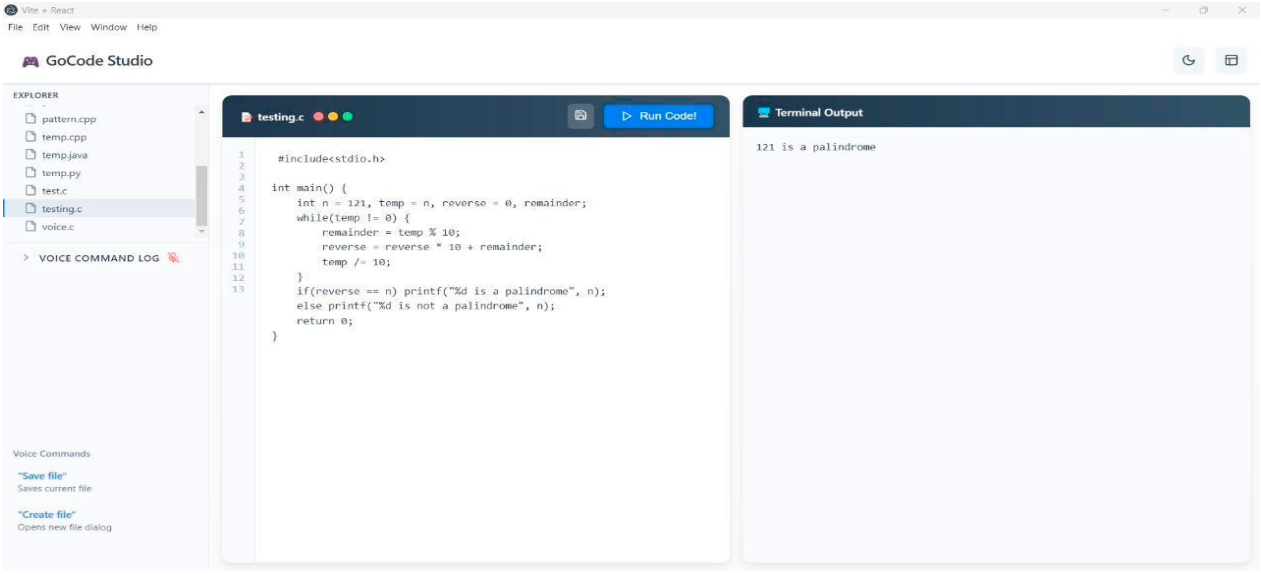

Fig 4. C program for palindrome checking written using voice commands with successful execution output

Fig 5 shows the system's ability to respond to user interface customization commands; the "Dark theme" voice command has changed the IDE's appearance successfully. The interface's voice command log, and command list at the bottom of the interface, provides continuous feedback and commands to the user, which is helpful for useability and ease-of-use, particularly for novice users.

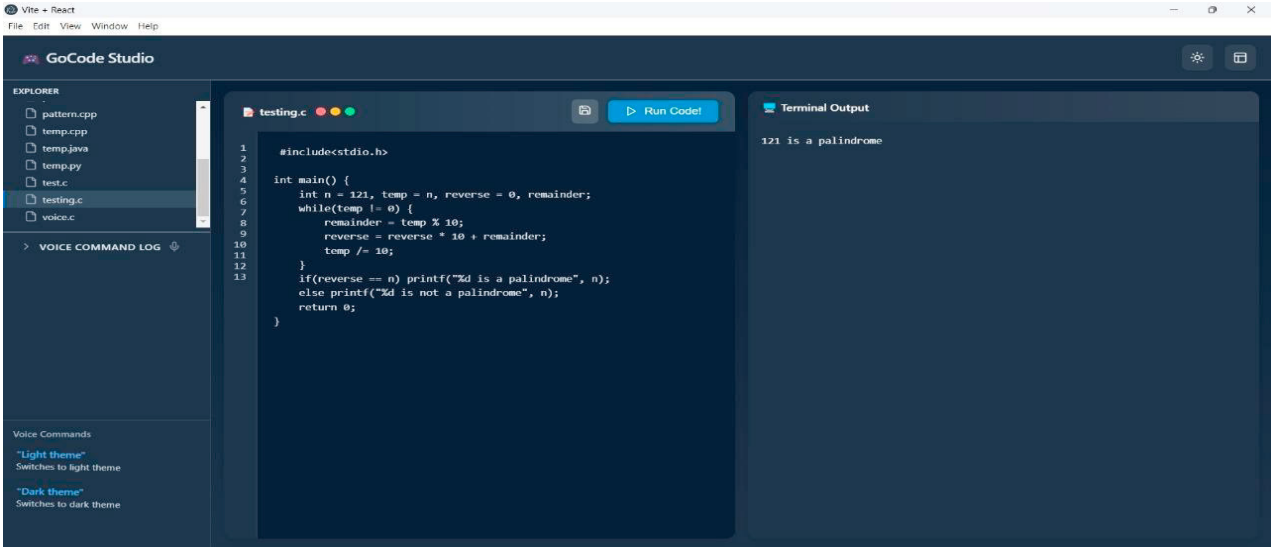

Fig 5. IDE appearance customization through voice command for dark theme

The implementation demonstrates several key functions that include: performing a file management task (making and saving files) using voice commands, dictating the code for Python while generating syntax correctly, compilation and execution of code, personalizing the interface, and ongoing feedback for the user using the voice command log. The screenshots support that the CodeWithDisability system accomplishes its main goal of providing completely hands-free programming, from file creation to code execution, allowing limbs to be free, for people with physical disabilities that may not allow the use of keyboard or mouse.

4.2. System Performance Evaluation

Based on our evaluation of the recognition module, approximately 80% of standard programming commands are recognized on the first attempt. Table 3 shows the recognition accuracy rates for various categories of voice commands.

Table 3. Voice Command Recognition Accuracy

Command Category	**Examples**	**No. Of Times Executed**	**No. of times executed Rate correctly**	**Accuracy**
Variable Declarations	“declare int x”	20	18	89%
Control statements	“if statement with x greater than zero"	27	22	80%
Function calls	“Call function calculateArea with arguments length, comma, width”	30	24	80%
File Operations	“save file”, “create new file”	25	23	92%
Code Execution	“compile program”, “execute”	28	25	90%

On average, the command processing time took 3 seconds from voice input to code generation, which is acceptable for real-world programming workflows. It allows for a comfortable coding experience without interfering too much with the user's thought process.

4.3. Technical Limitations

The success of the speech recognition system was inconsistent based on accent types and speech patterns and often required users to modify their speaking rate or pronunciation to yield success. This issue is especially relevant in a global context with programming occurring across many languages and dialects. This implementation utilizes python speech recognition packages without incorporating machine learning models that could potentially adjust the recognition capabilities for a variety of speech patterns. The current implementation offers an available function, but it does not provide the level of automation that may be obtained by introducing sophisticated speech recognition solutions. Findings from this inquiry indicate that there is a need for improved and tailored speech processing algorithms for programming contexts that can respond to an individual's speech patterns.

5. Conclusion

CodeWithDisability is a clear proof of feasible hands-free programming and has addressed a key issue in accessibility by the software development community. By integrating a specialized structure with a voice command processing system with predefined mapping to syntax, this research has contributed to an implementation that would permit people with physical disabilities to write code independently and efficiently. The demonstrated command recognition performance (80% accuracy) and acceptability of the coded output, confirms that voice programming can be practical as well as feasible for real world programming. CodeWithDisability also has a modular system so it can easily be extended for additional programming languages, not just C. This important characteristic will insure CodeWithDisability has a future as an assistive technology in the software development domain. The speech to coded mapping implemented here using structured datasets has been effective, and reasonable to expect could be adapted to other programming paradigms and languages. Future work should focus on increasing recognition accuracy for more complex programming constructs, reducing response time, and offering additional language support. Furthermore, integrating into IDE features like debugging tools and version control would improve functionality for more productive professional development contexts. CodeWithDisability presents opportunities for increased diversity and inclusion into technology through enabling people with physical disabilities to participate more fully in software development. This research has shown it is clear that with the right technologies that provide assistance, barriers to programming can be lowered significantly, allowing for new job and education pathways for people with disabilities.

6. Acknowledgement

We sincerely thank the Department of Computer Engineering at D.J. Sanghvi College of Engineering for providing all resources and infrastructure for carrying out this research. We would like to express our deep gratitude to Prof. Ruhina Karani for their advice, guidance, and support.

7. References

[1] Jeffrey M. Perkel, "Coding for Everyone", in Nature, Vol. 584, pp. 188-190, 2020.

[2] Julian Nowogrodzki, "Writing Code Out Loud", in Nature, Vol. 563, pp. 141-142, 2018.Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[3] David R. Flatla, Jayne B. Flick, Cecily Morrison, "Making Coding Education More Accessible for Blind and Visually Impaired Youth", in International Journal of Computer Science and Engineering Studies, Vol. 6, Issue 2, pp. 78-91, 2021.

[4] Susanna Gertsman, Andreas Holzinger, "Integrating Accessibility into Modern Software Development: Challenges and Opportunities", in Lecture Notes in Computer Science, Vol. 12765, pp. 134-149, 2021.

[5] Adnan Rahim Khan, Charith C Shetty, Deepak C A, Deepanshu Kumar Pali, Abhinav R B, "Programming using Voice for Physically Challenged Individuals", in International Journal of Advanced Research in Computer and Communication Engineering, Vol. 11, Issue 5, pp. 1-8, 2022. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[6] H. N. Kim, "Usability Assessment of Voice-Enabled Technologies for Users with Visual Disabilities," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 68, no. 1, pp. 1852–1854, 2024.

[7] Manuel B. Garcia, John Benedic R. Enriquez, Rossana T. Adao, Ari Happonen, "Hey IDE, Display Hello World": Integrating a Voice Coding Approach in Hands-on Computer Programming Activities, in International Conference on Computing Education and Research, Singapore, pp. 245-252, 2022.K. Elissa.

[8] Desilets, A. VoiceGrip: A Tool for Programming-by-Voice. *International Journal of Speech Technology* 4, 103–116 (2001).

[9] S. Nowrin and K. Vertanen, "Programming by Voice: Exploring User Preferences and Speaking Styles," in *Proceedings of the ACM Conference on Conversational User Interfaces (CUI '23)*, Eindhoven, Netherlands, July 19–21, 2023. ACM, New York, NY, USA

[10] S. C. Arnold, L. Mark, and J. Goldthwaite, "Programming by Voice: A System for People with Motor Impairments," in *Proceedings of the Fourth International ACM Conference on Assistive Technologies (ASSETS '00)*, Arlington, Virginia, USA, November 13–15, 2000, pp. 149–155.

[11] R. S. Maloku and B. Xh. Pllana, "HyperCode: Voice aided programming," in *IFAC-Papers On Line*, vol. 49, no. 29, pp. 263–268, 201.

[12] R. Jha, M. F. H. Fahim, M. A. M. Hassan, C. Rai, M. M. Islam and R. K. Sah, "Analyzing the Effectiveness of Voice-Based User Interfaces in Enhancing Accessibility in Human-Computer Interaction," *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*, Jabalpur, India, 2024, pp. 777-781.