VOICE BASED PROGRAMMING IDE USING ARTIFICIAL INTELLIGENCE

Jibin P Joseph, Amil Thomas, Anila Abraham, Parvathy.V

Students with a B Tech degree are 1, 2, 3, and 4, respectively.

[1,2,3,4]Computer Science and Engineering.

[1,2,3,4]Kottayam Institute of Technology and Science,Kottayam,India.

Abstract - This project presents a novel approach to programming by developing a voice-based programming IDE that utilizes artificial intelligence. The system aims to make programming more accessible and intuitive, allowing users to create, modify, or choose code from a pre-existing dataset by speaking natural language instructions. By utilizing machine learning and AI algorithms, the system generates code based on vocal inputs and a pre-existing dataset. This innovative approach has the potential to improve programming literacy and empower individuals with little to no coding experience to develop software. The project seeks to democratize programming and make it accessible to everyone, potentially revolutionizing the way people approach programming.

Index Terms - Integrated Development Environment (IDE), Google Text to Speech (GTS), Automated Speech Recognition (ASR), Voice Driven Device (VDD), Extended Local Ternary Pattern (ELTP) Latent Dirichlet Allocation (LDA), Augmented Reality (AR),Natural Language Processing (NLP), Recommended System (RS), Named Entity Recognition (NER), Part of Speech (POS).

I. INTRODUCTION (VOICE BASED PROGRAMMING IDE USING AI)

Software developers often perform their tasks in integrated development environments (IDEs), which integrate multiple tools, such as, advanced source code editors, testing tools, automatic compilers, and debuggers. It is well known that development tools affect software development efficiency and quality, by speeding up repetitive tasks and allowing programmers to focus on the more critical aspects of the process. IDE commands are menu items and keyboard shortcuts that carry out a variety of operations, including Save, Run, Open Resource, and others. IDE manufacturers and academics developed many forms of recommender systems (RS) to enhance command knowledge. In software engineering, RSs are defined as software applications that offer data deemed valuable for a software engineering task in a specific context. In fact, very frequently, in high-functionality applications, such as IDEs, the lack of adequate tool knowledge is the primary cause of potentially useful functionality not being used. Furthermore, as was already mentioned, tool knowledge has a direct impact on the productivity increase that the tool is intended to produce. Moreover, since IDEs are complex systems serving the needs of a large and diverse user population The voice-based programming IDE is about building program by vocal instruction. when programmer instruct to build a program by their own voice the IDE will convert voice to text according to that text it will generate code while compiling data-sets.

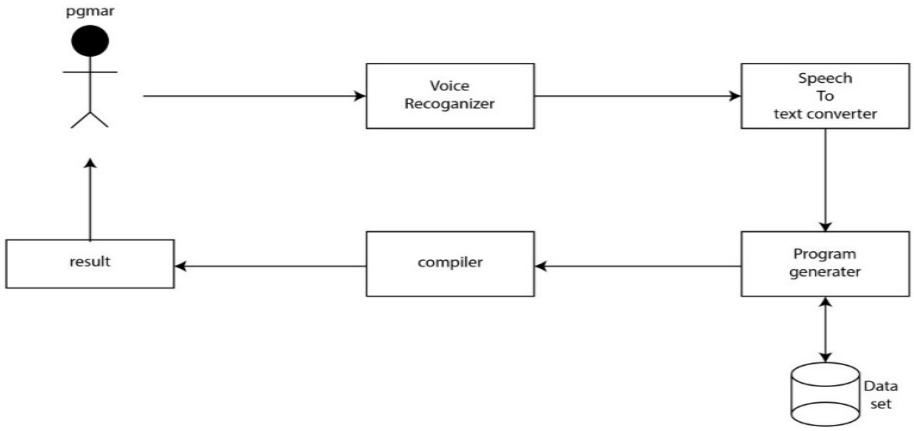
II. LITERATURE SURVEY

Santhosh B, Prajwal N. Srivatsa, and Subash S. [1] Voice control is a feature that is expanding and affecting how people live. Voice assistants based on AI recognize human voice and respond via integrated voices. In this project, Python programming language is used to build an AI-based voice assistant that converts audio to text and uses GTTS to convert text to English language audio files played using the play sound package. The voice assistant waits for a pause, sends requests to its database, and splits them into separate commands to understand the user's request. ASR systems are used to convert speech to text by recording speech, creating wave files, cleaning background noise, normalizing volume, breaking down elements, analyzing sequences, and implementing statistical probability. The voice assistant asks a question if it doesn't understand the request and performs the task if it understands. This project has implemented many features and is useful in various fields, including business and laboratory, where it can help those wearing gloves or body suits to obtain information hands-free. **Jiacheng Shang and Jie Wu[2]** The literature survey discusses the use of voice-based input in augmented reality (AR) headsets and the security concerns associated with it. Recent studies demonstrate that in systems without speech verification, attackers can insert inaudible voice instructions. To defend against such voice-spoofing attacks, the literature proposes a voice-spoofing defense system for AR headsets that leverages both the internal. Voices from people may be spread through the body and the air.The system can identify regular users with an average accuracy of 97% and fight against two common types of assaults with an average accuracy of at least 98%. Two classification models are proposed to detect voice input from attackers by targeting different attack models. The system can defend against obstruction assault and replay attack with average accuracy of 99.2% and 98%, respectively, according to experimental data. The proposed system is the first to protect voice input for AR headsets by measuring the correlation and similarity between the internal body voice and air voice. **Kostadin Damevski,Hui Chen,David C[3]**The literature survey discusses recent studies that show users of complex applications are accepting and utilizing command recommendations. The focus of this work is on improving the recommendation generation by modeling future task context to make better recommendations. Using Temporal Latent Dirichlet Allocation, a topic

model for natural language applied to software development interaction data, the suggested method forecasts future development instructions. This technique appears to be useful for forecasting future IDE commands and providing observations with an empirical interpretation, according to the evaluation on two sizable interaction datasets for several IDEs. The paper presents a novel approach to dimensionality reduction and future task context prediction using a time-sensitive variant of Latent Dirichlet Allocation (LDA). The technique addresses the noisy and time-based nature of IDE interactions and shows promise for analyzing data for recommendation systems in software engineering.

III.METHODOLOGY

Proposed System



3.1 Architecture of Proposed System

Figure 3.1 illustrates that when a programmer logs into the system, if the programmer wants to build a program. then the programmer will build the program with their own voices. The voice of the programmer will be recognized by the voice recognizer. After that, it converts speech to text by Using a speech-to-text converter. Then the program will generate from the dataset. then compiles it through the compiler and executes the program. After obtaining the required output, the user returns to the login page.

(1) Speech Recognition Module

Speech Recognition will take human vocal instructions as input and convert them into text. It will use pre-trained models or machine learning techniques to recognize speech accurately.

(2) Natural Language Processing (NLP)

Natural Language Processing will analyze the text output from the speech recognition module and extract the meaning of the instruction. It will use NLP techniques such as Named Entity Recognition (NER) and Part of Speech (POS) tagging to analyze the text.

(3) Code Generation

Code Generation will use the output from the NLP module to generate or modify the code. It will use machine learning techniques such as natural language processing or neural networks to train a model on a dataset of code snippets.

(4) Code Execution

Code Execution will be responsible for executing the generated code and providing output to the user. It will use a programming language such as Python or Java to implement this module.

(5) User Interface

This module will provide an interface for the user to interact with the system using voice commands. It will be designed to be user-friendly and intuitive, allowing users to create code quickly and easily using voice commands.

(6) Dataset

This module will provide the dataset of code snippets that the system will use to generate or modify code. It could use existing datasets such as GitHub or Stack Overflow or create its own dataset.

(7) Testing and Evaluation

This module will be responsible for testing and evaluating the performance of the system. It will use various metrics such as accuracy, efficiency, and usability to evaluate the system's performance.

IV.IMPLEMENTATION

It is a detailed breakdown of the methodology used to achieve the project goals and objectives. This section typically includes information on data collection and preparation, model development, integration of different systems, testing and refining the system, and any challenges encountered during the implementation process. The implementation section is critical as it provides a clear understanding of how the project was executed and how the final product was developed.

(1) Dataset Collection and Preparation

To build the system, a dataset of programming code snippets that can be used to build new programs is needed. The dataset can be collected from various sources such as online repositories or created from scratch. The dataset needs to be prepared, cleaned, and duplicates need to be removed.

(2) Dataset Pre-processing

The dataset needs to be preprocessed to make it usable by the system. The preprocessing can involve tasks such as cleaning the data, removing duplicates, and converting the code snippets into a format that can be easily processed by the system.

(3) AI Model Development

The AI model is the core of the system. An AI model that can process human vocal instructions and generate new programs based on the dataset needs to be developed. This would require training the AI model using the pre-processed dataset.

(4) Integration of AI Model and Voice Recognition System

Once the AI model is developed, it needs to be integrated with a voice recognition system. This can be done using a speech-to-text API that converts human vocal instructions into text that the AI model can process.

(5) IDE Development

The IDE is the user interface through which the user interacts with the system. An IDE that allows the user to input vocal instructions and view the generated code needs to be developed.

(6) Testing and Refining the System

The final step is to test the system to ensure it works as intended. The system may need to be refined by tweaking the AI model or improving the voice recognition system based on user feedback.

Overall, the implementation of the project requires expertise in programming, AI, voice recognition, and UI/UX design. These skills are essential to create a functional and user-friendly voice-based programming IDE.

III. CONCLUSIONS

In conclusion, the voice-based programming IDE using artificial intelligence is a promising project that offers a unique solution to the challenges of coding. The system's ability to generate code through voice commands can significantly reduce the time required for coding and make it more accessible to a wider range of individuals, including those with disabilities or limited typing proficiency. The project utilizes artificial intelligence to interpret human speech and generate code from a dataset of pre-existing code snippets or create new code based on user input. The user interface is user-friendly and intuitive, making the coding process more natural and intuitive. While the project shows great potential, there are areas for improvement that could enhance its functionality and usability. The integration of natural language processing, developing a larger code dataset, and integrating with other programming tools are just a few examples of future work that could enhance the system's performance and functionality.

Overall, the voice-based programming IDE using artificial intelligence is a significant step towards a more efficient and accessible coding environment. With further development and refinement, it could become an essential tool for developers looking to streamline their workflow and increase productivity.

IV. REFERENCES

[1] P. Bourque, and R. E. Fairley, The Guide to the Body of Knowledge in Software Engineering.. Washington, DC, USA: IEEE Computer Society Press, 2014.
[2] A Process-Driven Approach to Software Project Management by A.Ahmed Bangalore, India: Taylor & Francis, 2011.
[3] M. Robillard, R. Walker, and T. Zimmermann, ‘‘Recommendation systems for software engineering,’’ IEEE Softw., vol. 27, no. 4, pp. 80–86, Jul./Aug. 2010.
[4] T. Grossman, G. Fitzmaurice, and R. Attar, ‘‘A survey of software learn ability: Metrics, methodologies and guidelines,’’ in Proc. SIGCHI Conf. Hum. Factors Comput. Syst., 2009, pp. 649–658.
[5] E. Murphy-Hill, ‘‘Continuous social screen casting to facilitate software tool discovery,’’ in Proc. 34th Int. Conf. Softw. Eng., 2012, pp. 1317–1320.
[6] "User modelling in human-computer interaction," G.Fischer Customer Model. User-Adapted Interact., vol. 11, pp. 65–86, Mar. 2001.
[7] K. Damevski, H. Chen, D. C. Shepherd, N. A. Kraft, and L. Pollock ,‘‘ Predicting future developer behavior in the IDE using topic models ,’’ IEEE Trans. Softw. Eng., vol. 44, no. 11, pp. 1100–1111, Nov. 2018.
[8] M. Gasparic, T. Gurbanov, and F. Ricci, ‘‘Context-aware integrated development environment command recommender systems,’’ in Proc.32nd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE), Oct. 2017, pp. 688–693.
[9] E. Murphy-Hill, R. Murphy, Jiresal, "Improving software developers' fluency by proposing development environment commands," Proceedings of the 20th International ACM SIGSOFT Conference, vol. 2012, Symp. Found. Softw. Eng., pp. 1–11.
[10] M. Schmidmaier, Z. Han, T. Weber, Y. Liu, and H. Hußmann, ‘‘Real-time personalization in adaptive IDEs,’’ in Proc. Adjunct Publication 27th Conf. User Modelling, Personalization Adaptation, 2019, pp. 1-6.
[11] D. Silva, R. Terra, and M. T. Valente, ‘‘Recommending automated extract method refactorings,’’ in Proc. 22nd Int. Conf. Program Comprehension,2014, pp. 146–156.
[12] S. Zolaktaf and G. C. Murphy, ‘‘What to learn next: Recommending commands in a feature-rich environment,’’ in Proc. IEEE 14th Int. Conf. Mach. Learn. Appl., Dec. 2015, pp. 1038–1044