IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, November 2025
Impact Factor: 7.67

Realtime Web IDE with Code Recommendation and Voice Recognition

Hire Om Pundlik and Pawar Vaibhav Santosh
AIML Engineering
Loknete Gopinath Munde Institute of Engineering Education & Research (LoGMIEER), Nashik, India

Abstract: *A real-time Integrated Development Environment (IDE) that incorporates code recommendation and speech recognition features. The IDE is designed to assist developers by providing real-time code suggestions as they type, helping to reduce coding errors and increase efficiency. The speech recognition feature allows developers to dictate code, navigate through the codebase, and execute commands with ease, making coding accessible to individuals with physical disabilities or those who prefer a hands-free approach. Based on the current context and the developer's coding history, the IDE utilizes machine learning algorithms to recommend relevant code snippets.These suggestions are displayed in a user-friendly interface, allowing developers to easily choose and insert the recommended code into their projects. The speech recognition feature is powered by advanced natural language processing technology, enabling the IDE to accurately transcribe spoken commands and code. Overall, this real-time IDE with code recommendation and speech recognition is designed to improve developer productivity, reduce coding errors, and make coding more accessible for everyone.*

Keywords: Realtime, Integrated development environment, code recommendations, voice recommendations, voice interaction, code generation

I. INTRODUCTION

Integrated development environments (IDEs) often provide software developers with a platform where they can perform their tasks. These IDEs integrate multiple tools, such as advanced source code editors, testing tools, automatic compilers, and debuggers. It is well known that development tools play a significant role in influencing the efficiency and quality of software development. They help in speeding up repetitive tasks and enable programmers to focus on the more critical aspects of the process.

IDEs have yet to be fully supported by easy-to-use speech-to-text software, making coding inaccessible to those who struggle with or cannot use standard keyboards. Current coding practice implicitly relies on keyboard usage, which hinders individuals who cannot type on a keyboard from coding. However, there have been efforts to address this issue by building personal systems (such as Tavis Rudd's system and Ben Meyer's VoiceCode) for those willing to invest time in learning a new "language." IDE commands, which include shortcuts and menu buttons for functions like Save, Run, and Open Resource, are crucial for coding. To enhance command knowledge, IDE vendors and researchers have developed various recommender systems (RSs). RSs are software applications in the field of software engineering that provide valuable information in a specific context. For instance, they may suggest the use of a command that hasn't been utilized yet. In fact, inadequate tool knowledge often leads to underutilization of potentially useful functionality in highly functional applications like IDEs.

1.1 BACKGROUND

In recent years, the development of integrated development environments (IDEs) has seen remarkable advancements, driven by the increasing demands of modern software development practices. This research paper presents a novel approach to enhance the capabilities of web-based IDEs by integrating real-time code recommendation and speech recognition functionalities. The proposed system aims to improve developer productivity, code quality, and user experience by providing intelligent code suggestions based on context and leveraging speech recognition to enable

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, November 2025
Impact Factor: 7.67

hands-free coding. The paper discusses the design, implementation, and evaluation of the Real-Time Web IDE, highlighting its potential benefits and limitations. The results demonstrate the effectiveness of the system in assisting developers and streamlining the coding process.

II. LITERATURE REVIEW AND OBJECTIVE

- IDE Interaction Support With Command Recommender Systems (Journal IEEE Access)
  Date Of Publication: Year 2020 Methods Used: Real time database Machine learning Artificial intelligence Visualization Auto completion Native work Hands free coding Acceptable command recommendation
- VocalIDE : An IDE for Programming via Speech Recognition(ASSETS'17, Oct. 29–Nov. 1 2017, Baltimore, MD, USA) Context Color Editing, or CCE CCE will be expanded into more visual feedback vocal navigation selection tools
- CodeHelper: A Web-Based Lightweight IDE for E-Mentoring in Online Programming Courses (University of Glasgow Downloaded on August 17,2021 at 09:25:21 UTC from IEEE Xplore) A messenger module is developed for real-time code update and notification of code marking
- Recommendation System Development Based on Intelligent Search, NLP and Machine Learning Method Illia Balush, Victoria Vysotska and Solomiia Albota Lviv Polytechnic National University, S. Bandera street, 12, Lviv, 79013, Ukrain
- Aroma: Code Recommendation via Structural Code Search arXiv:1812.01158v4 [cs.SE] 17 Oct 2019 proc. ACM Program. Lang., Vol. 3, No. OOPSLA, Article 152. Publication date: October 2019
- Introduction to Various Algorithms of Speech Recognition: Hidden Markov Model, Dynamic Time Warping and Artificial Neural Networks 2014 IJEDR | Volume 2, Issue 4 | ISSN: 2321-9939

2.1 Objectives of the study

- To build a development environments that can reduce the complexity tasks in an IDE with simple recommendations.
- To provide a platform for developers to code hands free.
- To allow multiple developers to collaborate at the same time and helping out and correcting codes
- To display errors and provide their solutions then and there itself.
- To visualize the output in real time
- To allow user to save their work virtually using cloud without having to actually spare their device storage.

III. METHODOLOGY

3.1 HYPOTHEIS

In the context of this study, we defined two terms: how often recommendations were investigated and how often they were used after investigation. Our rationale is that recommendations cannot be helpful to a user if they are not investigated. Only when commands are used do the recommendations become elective and valuable to the user.

Investigation - We defined investigation as the user clicking on the recommendation to open the page containing detailed information about the command.

In our experiment, we designed it to investigate two hypotheses. Hypothesis 1 (H1) states that adding explanations to recommendations will result in more recommendations being investigated. Null Hypothesis 1 (H10) states that explanations do not affect the investigation of recommendations. We hypothesize that explanations enhance the users' inclination to investigate recommendations, especially in systems where recommendations are indirectly related to the user's task. This is supported by Sinha et al. [20], who argue that people are more likely to accept recommendations from their friends, and our previous findings that people learn more voluntarily from their peers [15, 16]. Regarding uptake, we defined it as the recommended command being used two or more times after investigating the recommendation

Based on our hypothesis, we expect that the Friend condition will be the most effective in improving the investigation of recommendations. Since our study involves novice to intermediate developers, we also predicted that participants would not be interested in the system's confidence in the recommendations due to a lack of understanding of how the recommender system functions. Therefore, we hypothesized that the Confidence condition would not be as effective as the other conditions.

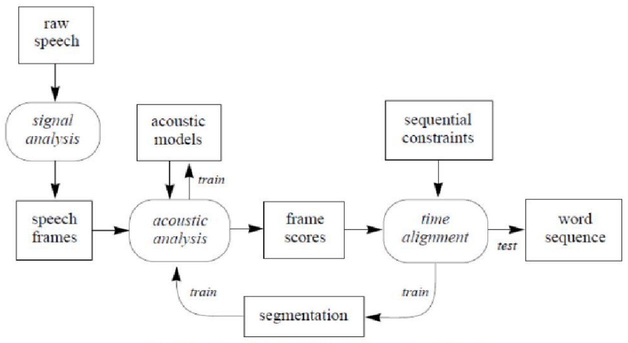


Fig. 1 Structure of standard Speech Recognition System

Fig 1: Structure of standard speech recognition system

3.2 Subtitle

Figure 1 illustrates the structure of a standard speech recognition system, consisting of the following elements:

1. Raw speech: Speech is typically sampled at a high frequency, such as 16 KHz over a microphone or 8 KHz over a telephone. This results in a sequence of amplitude values over time.

2. Signal analysis: The raw speech undergoes an initial transformation and compression process to simplify subsequent processing. Numerous signal analysis techniques are available for extracting useful features and compressing the data by a factor of ten without losing important information. Some popular techniques include:

- Fourier analysis (FFT): This technique provides discrete frequencies over time, which can be visualized. Frequencies are often distributed using a Mel scale, which is linear in the low range and logarithmic in the high range, reflecting the physiological characteristics of the human ear.
- Perceptual Linear Prediction (PLP): Inspired by physiological aspects, this technique yields coefficients that cannot be visually interpreted.
- Linear Predictive Coding (LPC): This technique produces coefficients of a linear equation that approximate the recent history of the raw speech values.

3.3 Algorithms

Code Recommendation Module (Generative Preprocessed Transformer) :

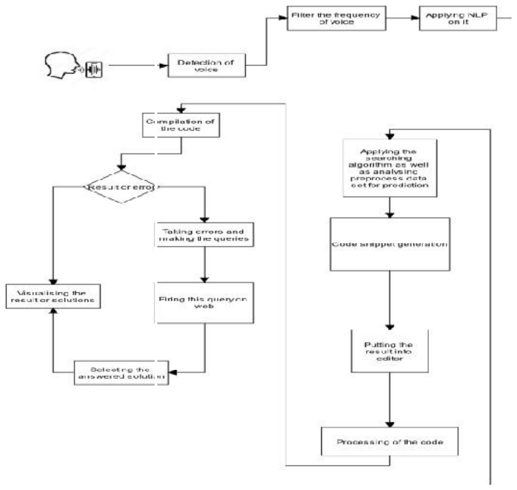
1. Preprocess the input text data by cleaning and tokenizing it into numerical input vectors.
2. Build the Generative Preprocessed Transformer model, which consists of multiple layers of self-attention and feed-forward neural networks.
3. Train the model on a large corpus of code data using self-supervised learning, where the model learns to predict the next word in a sequence of words given the previous words in the sequence.
4. Fine-tune the pre-trained model on a smaller corpus of text data for a specific natural language processing task, such as language translation or text classification.
5. Evaluate the quality of the generated text using metrics such as perplexity, which measures how well the model predicts the next word in the sequence, which measures how well the generated text matches a human-written reference text.

This project will be developed using web technologies and various Artificial intelligence algorithms

1. Speech API will be used for speech recognition
2. Technologies that will be used : React JS, Vanilla JS, Html, CSS, JSON, XML, NoSQL, NLP
3. Algorithms that will be used :
4. Searching – for implementing search function throughout the code
5. Map Reduce – AI related function works for coding block
6. Sorting – For the auto completion and formatting. Optimizing imports
7. Recursive Algorithm – For looping the formatting work.
8. Divide and Conquer- Functions
9. Backtracking- Folding codes and comments, functions, classes

3.3 MODULES

- Collaborative module : A socket is created which is used to map users of the system in one group for collaboration and this socket includes name and ID and data (code)
- Editor module : This is text area that contains code and the design of the code (body of the code). This text area beautifies code with it's syntax.
- compiler module: This module takes code from the editor module and sends to the API that can compile the code on that server and takes the input as a output of the sent code. this module also provides logs from the API. this module also provides status o the code (warnings, half compiled, full compiled, successful, timeout, failed etc)
- speech to text module: this is the speech recognition module that takes input from the microphone and converts this voice into text. this module also recognized various local as well as global languages and takes the input in those
- text to speech module: this module is used to give the instructions to the user as well as chat with user (talkback feature) for the interactive code development. this module also provides suggestive instructions (hints) to user while coding
- Authentication module: this module is provided by google with the help of firebase. this module can authenticate users with the help of email ID and password as well as log in with google, facebook, twitter or GitHub


Filter the frequency of voice
Applying NLP on it
Detection of voice
Compilation of the code
Result or error
Taking errors and making the queries
Visualising the result or solutions
Firing this query on web
Selecting the answered solution
Applying the searching algorithm as well as analysing preprocess data set for prediction
Code snippet generation
Putting the result into editor
Processing of the code

3.4 Architecture

The Realtime web IDE should be built using a microservices architecture, where each microservice handles a specific aspect of the IDE's functionality. This allows for easy scaling and maintenance of the application. The following microservices should be used:

User authentication and authorization service: This microservice handles user authentication and authorization. It should integrate with popular identity providers such as Google, Facebook, and GitHub.

Realtime collaboration service: This micro service handles real-time collaboration between multiple users working on the same project. It should provide real-time synchronization of changes made by users and should support multiple programming languages.

Code editor service: This microservice provides the code editor interface to the user. It should support multiple programming languages and provide features such as code highlighting, auto-completion, and debugging.

Code execution service: This microservice handles the execution of the code. It should support multiple programming languages and provide a sandboxed environment for running code.

Version control service: This microservice provides version control integration with popular version control systems such as Git and Subversion. It should allow users to manage code revisions, merge changes, and revert to previous versions. Algorithm

Realtime collaboration

The realtime collaboration service should implement an efficient algorithm for real-time synchronization of changes made by users. Here's a possible algorithm:

When a user starts editing code, the client sends a request to the server to obtain a unique lock on the file being edited.

When the server receives the request, it checks if the file is already locked by another user. If it is, the server responds with an error message. Otherwise, the server grants the lock and sends a notification to other users that the file is now locked.

The user edits the code and sends the changes to the server. The server updates the file and sends a notification to other users that the file has been updated.

When another user tries to edit the same file, the client sends a request to the server to obtain a lock. The server responds with an error message since the file is already locked.

When the original user finishes editing the file, the client sends a request to release the lock. The server updates the file and sends a notification to other users that the file is no longer locked.

Code execution

The code execution service should implement an efficient algorithm for running user code in a sandboxed environment.
The algorithm could possibly be like :
When a user submits code for execution, the client sends a request to the server to execute the code.
The server receives the request and starts a new sandboxed environment for executing the code.
The server compiles or interprets the code and executes it within the sandboxed environment.
The server captures the output of the code execution and sends it back to the client.
If the code execution encounters an error, the server captures the error message and sends it back to the client.

Version control

The version control service should implement an efficient algorithm for managing code revisions, merging changes, and reverting to previous versions. The algorithm can possibly be like :
When a user makes changes to code,
The client sends a request to the server to create a new commit.
The server receives the request

3.5 OVERVIEW

The architecture of the Real-Time Web IDE with Code Recommendation and Speech Recognition system is designed to facilitate seamless integration of code recommendation and speech recognition functionalities within a web-based IDE environment. The architecture consists of several key components that work together to provide an enhanced coding experience for developers.

1.1 Web IDE Component:

The Web IDE component serves as the foundation of the system, providing an interface for developers to write, edit, and execute code. It includes features such as syntax highlighting, code editor, file management, and collaboration capabilities. The Web IDE component interacts with other system modules to leverage code recommendation and speech recognition functionalities

1.2 Real-Time Code Recommendation Module:

The Real-Time Code Recommendation module is responsible for providing intelligent code suggestions to the developers as they write code. It analyzes the context of the code being written, including variables, functions, and libraries being used, to generate relevant recommendations. The module employs machine learning techniques, such as natural language processing and code analysis, to offer accurate and context-aware code suggestions. The module continuously updates the recommendations in real-time as the developer types

1.3 Speech Recognition Module:

The Speech Recognition module enables developers to interact with the IDE using voice commands. It converts spoken language into text, allowing developers to dictate code, execute commands, and navigate through the IDE hands-free. The module utilizes automatic speech recognition (ASR) algorithms and techniques to accurately transcribe spoken words into text. It also integrates with other system components to execute commands and perform actions based on voice inputs

1.4 Communication and Synchronization:

The architecture includes communication and synchronization mechanisms to ensure seamless coordination between the Web IDE component, Real-Time Code Recommendation module, and Speech Recognition module. Real-time updates and synchronization enable the IDE to provide live code recommendations as the developer types and respond to voice commands promptly. This communication may be achieved through API integration, message queues, or event-driven architectures.

1.5 Backend Services and APIs:

To support the functionality of the Real-Time Web IDE, backend services and APIs are utilized. These services handle tasks such as code analysis, natural language processing, machine learning model inference, and speech recognition. They provide the necessary infrastructure to process and analyze code snippets, generate code recommendations, and transcribe voice inputs accurately.

1.6 User Interface:

The user interface of the Real-Time Web IDE is designed to be intuitive and user-friendly. It incorporates visual cues and interactive elements to present code recommendations, highlight errors or warnings, and display transcription outputs from the speech recognition module. The user interface seamlessly integrates the code editor, code suggestion display, and voice command controls to provide a unified coding experience.

The overall architecture aims to provide a robust and efficient system that combines the power of real-time code recommendation and speech recognition within a web-based IDE environment. By leveraging these capabilities, developers can enhance their productivity, reduce coding errors, and achieve a more immersive and efficient coding experience.

IV. RESULTS AND DISCUSSION

For speech recognition

Participants were asked to direct a “human computer” (a researcher) to correct code that was provided to them. The participant's screen mirrored the display of the “human computer's” screen, who operated the text editor according to a

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, November 2025
Impact Factor: 7.67

very literal interpretation of participant's speech. During each of six levels, the participant was presented two blocks of code. The right of their screen displayed the correct and complete code, and the left displayed a similar code snippet that contained some errors. Participants were instructed to speak to the “human computer” to transform the incorrect code into the correct code. Errors in the code were listed to control for variation in debugging time and strategy
For command recommender
To evaluate the quality of IDE, assess its effect on user behavior, and study user interaction with the smart-IDE, we invited the students of the Introduction to Programming and Advanced Programming courses to participate to our study. The majority of the study participants installed the monitoring tools, listed below, when the Advanced Programming course began. Each user received recommendations from each algorithm for two consecutive weeks. The sequence of the algorithms was random, on the individual level, but we assured that there were only small differences between the overall numbers of participants who got recommendations in certain chronological order. Users had no information on how recommendations were generated.

V. CONCLUSIONS

The Smart IDE with code recommendation and voice recognition provides an approach towards a better hands free experience while coding, It can give solutions to various real world problems and its collaborative features can be proven effective for virtual classrooms or virtual coding tests where minimal code recommendation is required
It has a low learning curve which enables the users to interact with the system and hence providing a better user experience
Smart IDE attempts to solve real world problems in a given scope to justify the users needs and provide an optimal solution based on the particular problem

REFERENCES

[1]. 2013 Recommender System Using Collaborative Filtering Algorithm Ala Alluhaidan Grand Valley State University
[2]. P. Bourque, and R. E. Fairley, Guide to the Software Engineering Body of Knowledge. Washington, DC, USA: IEEE Computer Society Press, 2014.
[3]. Ahmed, Software Project Management: A Process-Driven Approach. Bengaluru, India: Taylor & Francis, 2011.
[4]. L. C. L. Kats, R. G. Vogelij, K. T. Kalleberg, and E. Visser, “Software development environments on the web", in Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software - Onward! ’12, (2012), pp. 99
[5]. M. Goldman, “Role-based interfaces for collaborative software development", in Proceedings of the 24th Annual ACM Symposium Adjunct on User Interface Software and Technology - UIST ’11 Adjunct, (2011), pp. 23.
[6]. F. Frößler, "A Practice Theoretical Analysis of Real Time Collaboration Technology: Skype and Sametime in Software Development Projects", Göttingen: Cuvillier, (2008).
[7]. S. Klein, N. Vehring, and M. Kramer, “Introducing Real Time Communication: Frames, Modes & Rules", in Proceedings 23nd Bled eConference eTrust: Implications for the Individual, (2010), pp. 591–606.
[8]. © 2014 IJEDR | Volume 2, Issue 4 | ISSN: 2321-9939 IJEDR1404035 International Journal of Engineering Development and Research (www.ijedr.org) 3590 Introduction to Various Algorithms of Speech Recognition: Hidden Markov Model, Dynamic Time Warping and Artificial Neural Networks Pahini A. Trivedi V.V.P. Engineering College Rajkot, Gujarat, India