



**IJIRCCCE**

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 7, July 2024

**ISSN** INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

**Impact Factor: 8.379**

 9940 572 462

 6381 907 438

 [ijircce@gmail.com](mailto:ijircce@gmail.com)

 [www.ijircce.com](http://www.ijircce.com)

# VSCODE: Code with Voice using Natural Language Processing (NLP)

Yuktha M A, Dr Sanjay Kumar C K

Student, Department of MCA, The National Institute of Engineering, Visvesvaraya Technological University, Mysuru  
Karnataka , India

Associate Professor & Head of the Department, Department of MCA, The National Institute of Engineering,  
Visvesvaraya Technological University, Mysuru , Karnataka , India

**ABSTRACT:** The project, titled "VSCODE - Code With Voice Using Natural Language Processing (NLP)," introduces an innovative approach to coding by enabling users to interact with the Visual Studio Code (VSCode) text editor entirely through voice commands. Leveraging advanced Natural Language Processing (NLP) techniques, the system transforms spoken language into text, interprets the user's programming instructions, and generates corresponding code snippets using a custom code generator API. The core functionality of the system lies in the seamless integration of speech-to-text conversion, NLP, and a dedicated code generation API within the VSCode environment. Users can articulate their coding instructions vocally, allowing for a hands-free coding experience. The NLP model is designed to comprehend a wide range of programming language constructs, syntax, and commands, ensuring accurate interpretation of user input. The project's architecture involves a robust Speech-to-Text (STT) module, which captures and transcribes spoken words into text data. The NLP component processes this textual input, extracting semantic meaning and intent related to programming tasks. Through a custom code generator API, the system translates these interpretations into coherent and syntactically correct code snippets tailored to the user's specifications. The Visual Studio Code extension serves as the interface for users, offering a collaborative environment where they can seamlessly integrate voice commands into their coding workflow. The extension includes features such as syntax highlighting, code formatting, and intelligent code completion to enhance the overall coding experience. By bridging the gap between natural language communication and programming, this project aims to empower users, particularly those with mobility challenges, to engage in efficient and expressive coding practices.

**KEYWORDS:** Natural language processing techniques , Google cloud speech to text API .

## I. INTRODUCTION

Natural language processing (NLP) technology has advanced significantly in recent years, making it possible for computers to comprehend and react to human language in ever-more complex ways. Not only has this advancement changed the way we communicate with digital assistants and search engines, but it has also created new opportunities to improve accessibility and productivity across a range of industries, including software development. Writing and running code typically calls for a high degree of technical expertise and knowledge of programming grammar and languages. However, this can be a major barrier to entrance for many people, particularly those who are new to coding or have little technical experience. Furthermore, it might be difficult to effectively debug code or articulate complicated coding principles, even for seasoned coders. Our suggestion is to create a project called "Code with VS Code using Natural Language Processing" in order to overcome these obstacles and improve programming's usability and accessibility. The goal of this project is to use natural language interactions (NLP) to write and run code, which will minimize the cognitive burden and learning curve that come with traditional coding methods. The main component of the project is a web application built with Flask that acts as the user interface for communicating with the system. Users can choose their preferred programming language, including well-known languages like Python, Java, and JavaScript, and give voice input using this application. Furthermore, ChatGPT's automated code generation feature allows users to select between manual code entry and NLP-assisted code entry. The project intends to democratize coding by enabling people to convey their programming ideas in a way that seems intuitive and natural by integrating voice input with NLP capabilities. Users can express their coding requirements in plain language, either through conversational interactions using ChatGPT or through manual input enhanced by NLP approaches. This allows users to bridge the gap between human intent and instructions that can be executed by a computer.

## II. LITERATURE SURVEY

The referenced works explore various aspects of using Natural Language Processing (NLP) and Artificial Intelligence (AI) for voice-activated systems, emphasizing their application in coding and automation. Key insights include the use of NLP models like BERT, CodeBERT, and Seq2Seq for understanding and generating programming code from voice commands, as seen in projects like "VSCODE - Code With Voice Using NLP." Studies highlight methodologies such as using RNNs, LSTM networks, and speech recognition technologies (e.g., DeepSpeech) to convert spoken language into executable code snippets. The integration of NLP with IDEs like Visual Studio Code aims to enhance accessibility and efficiency in programming, especially for users with mobility issues. Comprehensive testing, evaluation, and documentation are underscored as critical for developing effective voice-activated coding platforms. These works collectively emphasize the synergy between voice recognition, NLP, and AI in creating innovative, user-friendly coding and automation solutions.

## III. EXISTING SYSTEM

Writing and running code in the current system usually requires direct manipulation of programming languages and grammar, which can be intimidating for novice and inexperienced engineers. Even with features like syntax highlighting and code completion, Integrated Development Environments (IDEs) like Visual Studio Code (VS Code) still require users to have a basic understanding of programming syntax and ideas.

## IV. PROPOSED METHODOLOGY

### A. Proposed System:

The "Code with VS Code using Natural Language Processing" system that is being presented provides a new method of authoring and running code by incorporating NLP technology into the coding process. Users can communicate with the system utilizing voice input and natural language descriptions of their coding requirements using a Flask-based web application. The system provides two options for code generation: automated code generation via ChatGPT and human code input enhanced by natural language processing (NLP) methods. The system supports a number of programming languages, including Python, Java, and JavaScript.

The "Code with VS Code using Natural Language Processing" system offers numerous benefits that enhance the coding experience by making it more accessible, efficient, and user-friendly. By allowing users to input code via voice and natural language descriptions, the system increases accessibility, particularly for individuals with physical disabilities. It streamlines the coding process, enabling quicker code generation through natural language descriptions, which reduces the time spent on manual coding and debugging. This system is especially beneficial for beginners or those unfamiliar with specific programming languages, as it translates plain language descriptions into code, making the coding process easier to understand and execute. Supporting multiple programming languages like Python, Java, and JavaScript, the system offers versatility and flexibility, allowing users to switch between languages effortlessly based on project requirements. Automated code generation via ChatGPT helps reduce syntactical and logical errors, ensuring the production of correct and optimized code. Furthermore, the system enhances learning and skill development by providing interactive feedback on human code input enhanced with NLP methods. It also facilitates better collaboration among team members, including those without strong programming backgrounds, by allowing them to contribute through natural language descriptions. The ease of generating and modifying code through natural language accelerates the prototyping phase, enabling rapid testing and iteration of ideas. Integration with the popular VS Code environment ensures access to a robust set of development tools and extensions, boosting overall productivity. Lastly, incorporating NLP technology into the coding process offers a novel and intuitive user experience, making the coding process more interactive and engaging.

### B. Methodology:

To develop a robust backend system using the most recent stable version of Python, begin by installing Python and the Flask framework using pip. Setting up a virtual environment is crucial for handling dependencies and maintaining project isolation, ensuring that all packages are installed within this environment. After activating the virtual environment, install essential libraries like Python-Dotenv for environment variable management, Flask for handling HTTP requests, and the requests library for managing external requests. Configure Flask application settings to manage different configurations and protect sensitive data using environment variables.

Next, create a structured Flask application, including necessary directories such as static for CSS, JavaScript, and image files, and templates for HTML pages. Define routes and endpoints by mapping URL paths to view functions

with decorators to manage front-end HTTP requests effectively. Implement basic error and exception handling procedures to manage and log errors gracefully. Integrate Google Cloud's Speech-to-Text API to handle voice input, enabling users to express their coding needs verbally. Develop modules to capture audio input from users and forward it to the Speech-to-Text API, handling transcription responses by extracting relevant information for further processing.

Utilize NLP libraries like spaCy or NLTK to implement NLP algorithms, creating functions for tokenization, lexer, parser, and stemming to process natural language inputs. Translate user intents into executable code snippets using NLP techniques. Integrate OpenAI's GPT models to generate code automatically based on user-provided natural language descriptions. Develop interaction modules to communicate with the ChatGPT API, generating code fragments in real-time and formatting them for further processing and implementation.

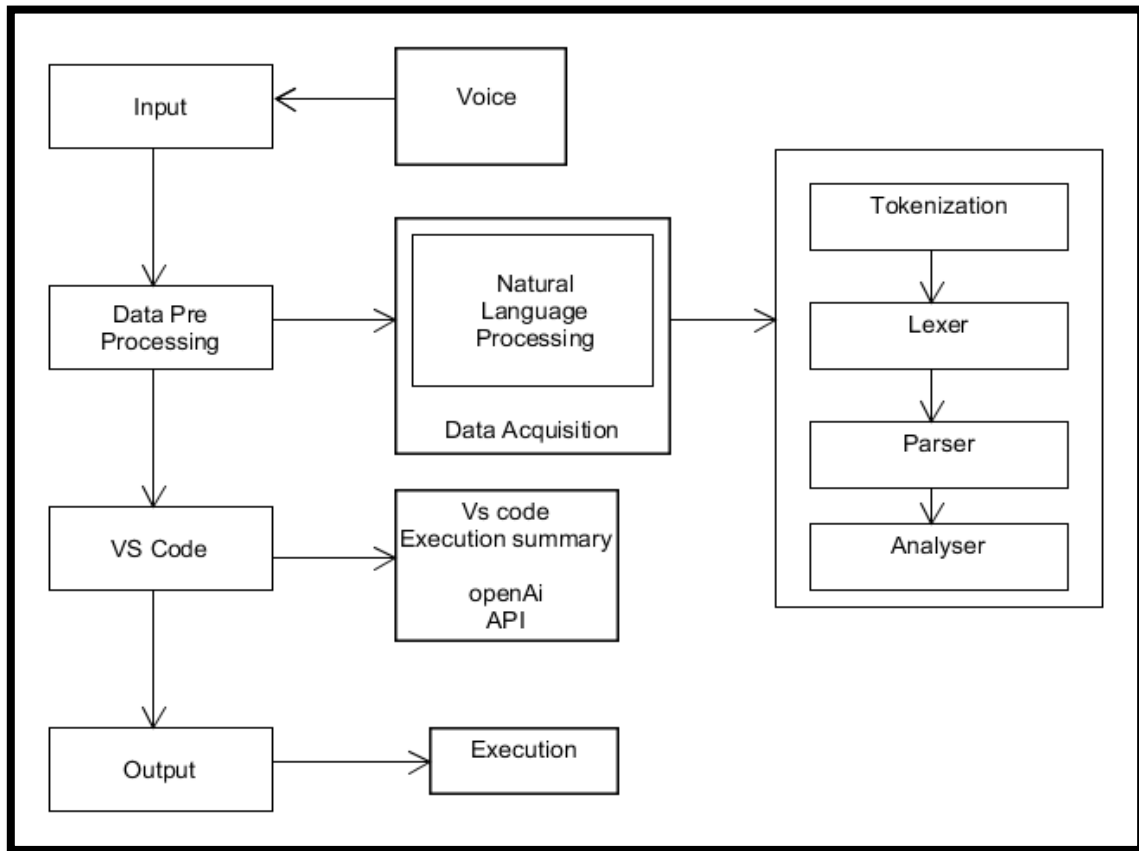
Implement functionality for manual code input, allowing users to describe code in plain language, which is then parsed and interpreted using NLP techniques to generate executable code. Create a custom VS Code extension using the VS Code extension development framework, developing modules to execute generated code snippets within the Visual Studio Code environment. Capture and display the output of code execution within the web application interface.

Finally, design and develop a user-friendly web interface using HTML, CSS, and JavaScript. Create forms and input fields for users to provide voice inputs, select their preferred language, and access code generation options. Incorporate interactive elements such as text fields, dropdown menus, and buttons to facilitate seamless user interaction.

Setting up the development environment for a project involves several steps. First, install Python and the Flask framework, ensuring you have the latest stable Python version for backend development, and use pip to install Flask. Next, create a virtual environment to manage dependencies and maintain project isolation, activating it to ensure all packages are installed within it. Necessary libraries like Python-Dotenv for environment management, Flask for handling HTTP requests, and requests for handling HTTP requests are then installed. Configure Flask application settings to manage various configurations and protect sensitive data using environment variables. Create the Flask application structure with directories for static files and templates for HTML pages. Define routes and endpoints using decorators to map URL paths to view functions, and implement error handling for managing and logging errors gracefully. For voice input processing, integrate Google Cloud's Speech-to-Text API to handle voice input, allowing users to verbally express coding needs, and develop modules to capture audio input and handle transcription responses. Implement NLP algorithms using libraries like spaCy or NLTK to process natural language inputs, developing functions for tokenization, parsing, and stemming, and translating user intents into executable code snippets. Integrate OpenAI's GPT models to automatically generate code based on natural language descriptions, developing interaction modules to communicate with the ChatGPT API, and format generated code for further processing. Implement manual code input functionality, allowing users to enter code descriptions in plain language, and generate executable code from these descriptions using NLP techniques. Integrate with VS Code by creating a custom extension for real-time code execution, developing modules to execute generated code snippets within the Visual Studio Code environment, and capturing and displaying output in the web application interface. Finally, design and develop a user-friendly web interface using HTML, CSS, and JavaScript, creating forms and input fields for voice input and code generation options, and implementing interactive elements like text fields, dropdown menus, and buttons for smooth user interaction.

## V. SYSTEM ARCHITECTURE

The "Code with VS Code using Natural Language Processing" solution is built on a micro services architecture, utilizing Python and a NoSQL database. It incorporates key technologies such as APIs and cloud services, comprising major components like the user interface, business logic layer, and data access layer. The design meets functional and non-functional criteria from system analysis, ensuring scalability, security, and usability. Its modular, adaptable, and easy-to-maintain structure allows for future improvements. The system enables voice commands for code execution and input within Visual Studio Code by integrating advanced NLP capabilities with a user-friendly web interface, coordinated by a Flask application. Key components include a Voice Input Processing Module using Google Cloud Speech-to-Text API, an NLP Module for interpreting user intentions, a Code Generation Module leveraging OpenAI's GPT models, and an Execution Module for running code snippets in VS Code. This combination enhances accessibility and efficiency in the development process.



**Fig 1 System Architecture**

The image depicts a system for understanding and executing natural language instructions

- **Input:** This box represents the initial input to the system, which could be spoken or written natural language.
- **Voice:** If the input is spoken, this box represents the process of converting the speech into text.
- **Data Pre-Processing:** This box represents the initial processing of the input text, which may involve tasks like cleaning the text, removing punctuation, and converting words to lowercase.
- **Natural Language Processing:** This is the core of the system, where the text input is analyzed and understood. It may involve tasks like identifying the subject, verb, and object of the sentence, as well as understanding the context and intent of the user.
- **Data Acquisition:** This box represents the process of gathering any additional data that may be needed to complete the instruction. For example, if the user asks to "play a song," this step may involve searching for the song in a music library.
- **Tokenization:** This box represents the process of breaking down the text into individual words or tokens.
- **Lexer:** This box analyzes the tokens to identify their grammatical categories, such as nouns, verbs, and adjectives.
- **Parser:** This box combines the tokens and their grammatical categories to create a structured representation of the sentence, known as a parse tree.
- **Analyser:** This box analyzes the parse tree to understand the user's intention and create a plan for executing the instruction.
- **VS Code:** This box represents a code editor where the instruction is translated into code. For example, a request to "create a new file" might result in code that creates a new file with a specific name and extension.
- **Vs code Execution summary:** This box represents the process of executing the generated code, which may involve using APIs or external libraries.
- **openAi API:** This box represents a specific API that is used to execute the generated code.
- **Execution:** This box represents the execution of the instruction based on the generated code.
- **Output:** This box represents the final output of the system, which could be a document, a song, or any other outcome based on the user's instruction.

## VI. RESULTS

### Homepage :

A system for processing natural language to generate code. The system starts with an input, which could be a voice command or written text. The input is then pre-processed, which may involve converting the input to text or removing unnecessary characters. The pre-processed text is then passed to a natural language processing module, which analyzes the text and extracts the meaning. This meaning is then used to generate code in the VS Code editor. The generated code is then executed and the output is displayed.

The natural language processing module uses a number of techniques to process the text. These techniques include:

- Tokenization: This involves breaking down the text into individual words or symbols.
- Lexing: This involves identifying the parts of speech of each word.
- Parsing: This involves analyzing the grammatical structure of the text.
- Analysis: This involves determining the meaning of the text.

Once the text has been processed, the system can generate code. This code is then executed and the output is displayed. This system is useful for automating tasks that are normally done manually. For example, it could be used to generate code for a web application based on a natural language description of the application.

### Time duration :

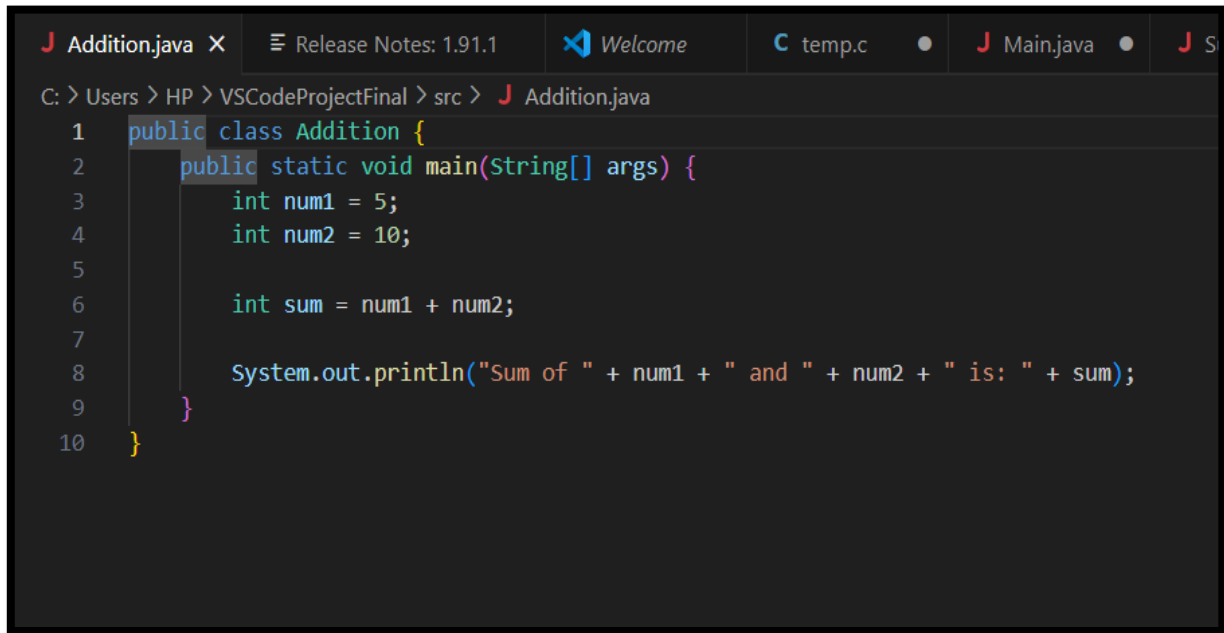
The diagram depicts the flow of information in a system that processes natural language input to generate code. The process starts with an input, which could be in the form of voice or text. This input undergoes data pre-processing to prepare it for natural language processing. The processed data is then fed into a natural language processing module, which utilizes various techniques to understand the meaning and intent of the input. The output of this module is then converted into code using a tokenizer, lexer, parser, and analyzer. The resulting code is then executed and the output is displayed. The diagram also highlights the use of a VS Code editor and openAI API for code development and execution. Overall, the diagram showcases a system that bridges the gap between natural language and code generation, facilitating automated code creation from human instructions.

### Selecting programming language :

It illustrates a workflow of a system that uses natural language processing (NLP) to convert voice input into code. The process begins with a voice input that is then converted into text. This text undergoes data preprocessing and is sent to a Natural Language Processing module, which is responsible for understanding the natural language input. The NLP module then uses tokenization, lexical analysis, parsing, and code analysis to convert the input into a format that can be understood by a code editor. Finally, the output is generated in the form of executable code, which can then be executed to generate the desired output. This workflow highlights the integration of NLP, code generation, and execution, showcasing a process that translates voice commands into tangible code.

### Selecting Method :

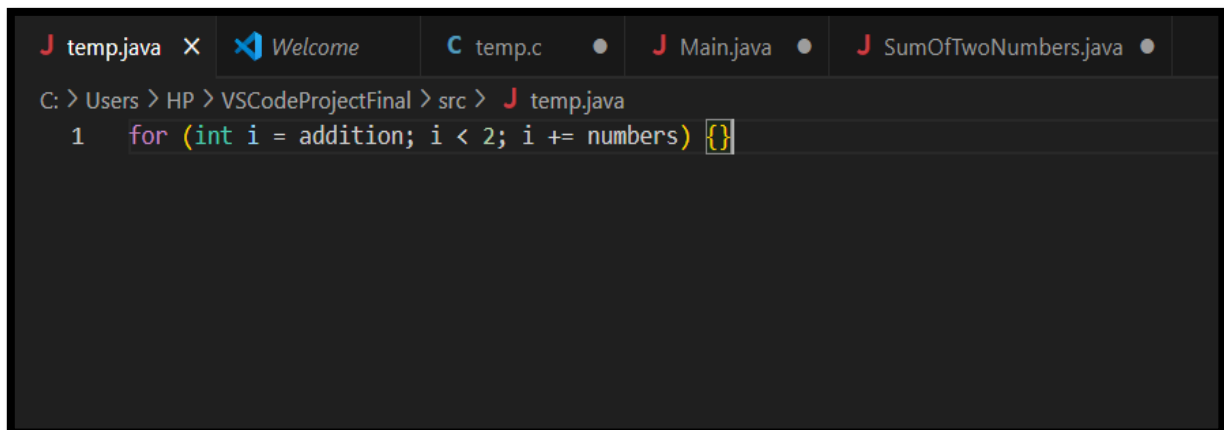
The screenshot shows a flow chart depicting a process that converts voice input into executable code. The process begins with voice input and undergoes data pre-processing to prepare it for natural language processing. This processed input is then sent through a series of steps: tokenization, lexical analysis, parsing, and analysis. These steps convert the input into a format suitable for code generation. Finally, the processed data is sent to VS code, where it is used to generate executable code. This code can then be executed to produce an output. The process utilizes an openAI API for code execution and summarization. Overall, this flowchart outlines a comprehensive process for converting voice input into executable code, leveraging natural language processing and code generation technologies



```

J Addition.java X  Release Notes: 1.91.1  Welcome  temp.c  Main.java  S
C: > Users > HP > VSCodeProjectFinal > src > Addition.java
1  public class Addition {
2      public static void main(String[] args) {
3          int num1 = 5;
4          int num2 = 10;
5
6          int sum = num1 + num2;
7
8          System.out.println("Sum of " + num1 + " and " + num2 + " is: " + sum);
9      }
10 }
    
```

Fig 2 program for selecting the Gen AI



```

J temp.java X  Welcome  temp.c  Main.java  SumOfTwoNumbers.java
C: > Users > HP > VSCodeProjectFinal > src > temp.java
1  for (int i = addition; i < 2; i += numbers) {}
    
```

Fig 3 program for selecting the Syntax library

## VII. CONCLUSION

By combining voice instructions and natural language processing techniques, the "Code with VS Code using Natural Language Processing" project has made substantial progress toward streamlining code input and execution. By utilizing OpenAI's GPT models and Google Cloud Speech-to-Text API, the solution facilitates natural language interactions between users and the VS Code environment, hence optimizing coding efficiency and productivity. Easy access to the system's features is ensured by the development of a user-friendly web interface, and real-time code execution and instant feedback are provided by the execution module. Even while the project has effectively automated the creation and execution of code, on going work is still required to improve the precision and resilience of NLP algorithms and to support other programming languages. Notwithstanding these difficulties, the project represents a noteworthy development in closing the gap between programming and natural language communication, providing promising prospects for future innovation and development.

### REFERENCES

- [1] Analysis of Machine Code Using Natural Language Processing N. Khurpiaa (2021)
- [2] Automation using Artificial intelligence based Natural Language Processing P. Mohana, M. Muthuvinayagam, P. Umasankar, T. Muthumanickam (2022)
- [3] Voice Assistant Using Artificial Intelligence Indukuri Manikanta Sai Varma, Kalidindi Pranith Varma (2022)
- [4] Designing of a Voice-Based Programming IDE for Source Code Generation: A Machine Learning Approach A. R. M. Nizzad and S. Thelijjagoda (2022)
- [5] Code Generator based on Voice Command for Multiple Programming Language S. Hossain, M. A. Emi, M. H. Mishu, R. Zannat, Ohidujjaman (2021)
- [6] Speech coding and speech recognition technologies: a review A. S. Spanias and F. H. Wu (1991)
- [7] VSCODE- Code With Voice Using Natural Language Processing(NLP) M. G. Prakash, A. Ponmalar, S. Deeba, A. Akilandeswari, S. B. M. Rasool, P. Lavanya (2022)
- [8] Voice controlled automation system M. S. Haleem (2008)





INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA



# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

 9940 572 462  6381 907 438  [ijircce@gmail.com](mailto:ijircce@gmail.com)



[www.ijircce.com](http://www.ijircce.com)

Scan to save the contact details