An AI-Powered Developer Assistant for Code Generation, Debugging and Research Automation

Miss **Sai Kiran Khandagale** Information Technology Department KBP Polytechnic Satara, India Email: saikhandagale45@gmail.com

Miss **Tanvi Dilip Shedage** Information Technology Department KBP Polytechnic Satara, India Email: tanvishedage780@gmail.com

Miss **Anuradha Vishnu Posugade** Information Technology Department KBP Polytechnic Satara, India
Email: Posugadeanuradha@gmail.com

Miss **Veena Kiran Kadale** Information Technology Department KBP Polytechnic Satara, India Email: veenakadale33@gmail.com

Under the Guidance of **Prof. Mrs. Patil M. M**
Information Technology Department
KBP Polytechnic, Satara, India
Email: mohinipatilkbp@gmail.com

Abstract

Software development is the collection of several software production tools requiring some special skills, such as programming language, debugging techniques, and the documents. Meanwhile, novice programmers may have difficulty to write bug free program, figure out and fix bugs, understand documents in technical language.

Recently, artificial intelligence techniques have been relatively successfully implemented to assist the developer, particularly in automating her repetitive programming task and improving programming efficiency. This paper proposed an AI-enabled developer assistant that provides several intelligent modules to simplify the development process of this software. Six modules built-in the developer assistant are User Management and Dashboard, Vibe Coding Assistant, AI Debug Mate, research and documentation assistant, Text-to-UI generator, and Voice-to-Text program generator.

All modules operate based on the natural language processing and generative AI techniques to instantiate human instruction language to computer program, detect and fix computer bug, extract key ideas of research website, and generate user interface automatically based on command text, respectively. Modular design offers great extensibility and flexibility with a systematic development procedure. Experiments verified the effectiveness of this engine in accelerating programmers task including programming and debugging, and research documentation.

Keywords

Artificial Intelligence, Code Generation, Natural Language Processing, Automated Debugging, Developer Assistant, Generative AI

1. Introduction

As new programming techniques become more complex, the need to use more advanced programming environment tools becomes vital. Programming today is much more complicated than it was a few years ago as developers have to perform many hard activities (source code write, code debugging, user interface designing, automatically generated documentation writing) which require considerable human effort. These activities are usually

very hard and time-consuming, especially for beginners who may miss some knowledge about debugging, designing or optimizing algorithms or understanding existing complex source code structures. That is why demand on such programming tools is high and programmers want to make their programming activities easier. Recent progress in artificial intelligence has brought a lot of changes in the programming process. Such tools as source-code writing assistants are more and more present which support programmers in a more effective source code writing. These systems are all about interpreting and transforming user's commands in natural language into a code which could run under computer.

If human language could be connected with programming languages, most of programmers' pains of learning complicated code syntax could be eliminated while focusing on solving more logical problems. Human language is one of the most significant one in human-computer communication

domain. The goal of natural language processing (NLP) is to interpret users provided textual commands and automatically convert it into code, used for execution. When combined with machine learning and other AI techniques, modern NLP systems are able to create not only source code but also describe already existing code structures and discover bugs in source. All these features had an enormous impact on source code writing and debugging every day. Also, they are most useful for beginners. The process of finding bug in a source code is also very hard and time-consuming, so intelligent debugging agent would analyse source structural information and error message in order to anticipate the source of bugs and suggest the changes to fix it. In addition to sources creation and debugging, modern integrated development environments (IDEs) allow for an easier programmer's experience through significant increase of accessibility programming environment tools. Voice Control user interface and research assistants for scientific papers writing are created.

Voice-to-text programming system enables developers to give verbal instructions to be converted to source code. Research assistant is able to summarize scientific articles and list the main points on it so that the user can more easily understand the scientific issue. All these features together integrated into one developer platform would modernize traditional programming environment.

The fast source-to-source translation, debugging agent, scientific papers summaries and GUI generator could be implemented in one advanced environment. This environment would give programmers more efficient and comfortable work. The goal of this project is to develop modular development environment supporting various activities in the software development process, based on artificial intelligence methods.

Table 1: Comparison of Existing Systems with the Proposed System

System / Research Work	Technology Used	Key Features	Limitations
NL2Code Framework	Natural Language Processing + Machine Learning	Converts natural language instructions into programming code	Supports only limited programming languages
Natural Language to Python Code Systems	NLP + Compiler Techniques	Generates Python programs from textual descriptions	Mostly restricted to Python programming
Transformer-based Code Generation	Deep Learning /	Provides code suggestions and auto-completion features	Requires large datasets for training
AI Code Debugging Assistant	Static Code Analysis + Machine Learning	Detects syntax errors and suggests possible fixes	Less effective for complex logical bugs
AI Programming Assistants	Generative AI Models	Helps developers with code suggestions and documentation	Primarily focused on coding assistance
Research Paper Summarization Tools	NLP + Text Summarization	Extracts key ideas and summaries from research documents	Not directly connected with development tools
Voice-to-Code Systems	Speech Recognition + NLP	Converts spoken commands into programming instructions	Accuracy depends on voice recognition quality
GUI Generator Tools	Text-to-UI Frameworks	Automatically creates basic user interface layouts	Limited flexibility in design customization
Integrated AI Development Tools	AI + IDE Integration	Combines coding assistance with development environments	Provides only a limited set of development features
Proposed AI- Powered Developer Assistant	Generative AI + NLP + Machine Learning	Integrates code generation, AI debugging, research summarization, Text-to-UI generation and voice-to-text programming in a single platform	Future improvements required for supporting more languages and advanced AI models

Table 1 provides a comparison between several existing AI-based tools used in software development and the proposed AI-powered developer assistant. Many of the existing systems focus on a single capability such as code generation, debugging support, or research document summarization. However, the proposed system combines multiple intelligent modules within one integrated platform. It supports natural language code generation, automated debugging assistance, research summarization, user interface generation from text descriptions, and voice- based programming. By bringing these features together, the system aims to simplify the development process and help programmers' complete common tasks more efficiently.

2. Literature Review

Many studies have looked at how artificial intelligence can help in making software. Research shows that

models that learn from data can turn text descriptions into code. The NL2Code system uses a mix of natural

language tools and coding knowledge to create working code from word Another study uses compiler-based

methods to turn natural language into Python code, focusing on how language structure matches programming

parts. Generative AI models are also common in software work. Transformer-based models are good at making

relevant code snippets and helping programmers.

These models learn from large code collections, find patterns, and give suggestions. Besides creating code, AI

tools are built to find and fix errors automatically. They look at code syntax and run-time messages to spot

issues and suggest fixes.

This new research adds to this work by combining many AI tools into one platform. Instead of just making code

or fixing bugs, the system offers several smart features to help developers do more tasks.

3. Proposed System

The system is planned as a modular AI developer assistant that combines various modules on a single platform.

These modules highlight different areas of the development process and share a common backend.

Module 1: User Management and Dashboard

This module gives the authentication and access control features to the application. Users are able to create

accounts, login the system with the registered user details. Managed access to all feature by the main

dashboard screen.

Module 2: Vibe Coding Assistant

The Vibe Coding Assistant allows programmers to express the program they wish to write or its behaviour in

natural language. Then the Assistant translates that textual description into programming language.

Module 3: AI Debug Mate

This module looks for erroneous code and provides possible solutions by correcting errors. It searches for

syntax errors and error messages in the code to identify likely points of failure in order to suggest more suitable

code solutions.

Module 4: Research and Documentation Assistant

The research module enables users to upload research papers or document files which will then be summarised

by the system and important information extracted that will be useful for software development.

Module 5: Text-To-UI-Generator

User interterface (UI) design can be a strenuous task; the Text-to-UI system allows developers to input text

descriptions of interfaces, automatically converting them into HTML or component-based graphical user

interface interfaces.

Module 6: Voice-to-Text Programming

This module gives voice enabled interaction to the development assistant. Spoken commands are transcribed

and sent to the Development assistant which in turn processes text and outputs program code files or performs

other actions.

4. System Architecture

The proposed system follows a layered architecture consisting of four main components: User Interface

Layer

Application Processing Layer AI Model

Integration Layer Data Storage Layer

The user interface layer provides the user interaction with the system using web- accessed applications. The

application layer takes responsibilities of the system logic and the modules communications. The AI models

modules are responsible for the application tasks such as code generation, debugging analysis, document

summarization. The data storing layer stores user data and system information.

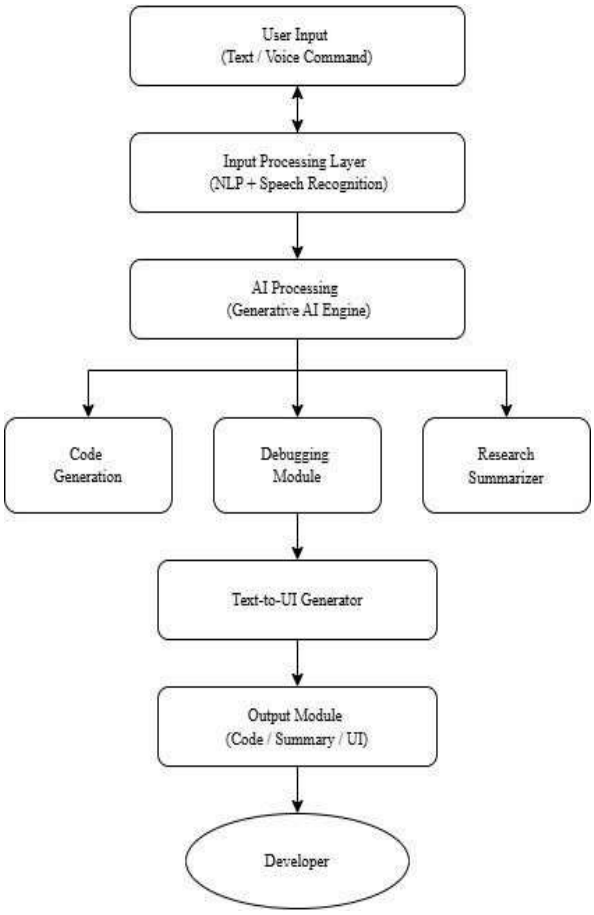

Figure 1: System Architecture of the Proposed AI-Powered Developer Assistant

Figure 1 illustrates the architecture of the proposed AI-Powered Developer Assistant. The system begins by processing and speech recognition techniques to understand the user's request. The processed information is then

passed to the AI processing engine which uses generative AI and machine learning models to analyze the task.

Based on the request, the system activates different modules such as code generation, debugging assistance, and

research summarization. The text-to-UI generator is used to automatically create basic user interface structures

from textual descriptions. Finally, the output module delivers the generated results such as program code,

summarized information, or interface layouts to the developer.

5. Results and Discussion

The demonstrated benefits of the prototype system are the ability to work with multiple different AI-assisted

programming tools in a single workspace. The code generator is able to produce basic code based on

commands given in natural language and the debugger was able to recognize common syntax errors and give

suggestions for fixes.

You can browse through the articles summarized in the image folders, so it is easier for you to understand the

technical information. Text-to-UI the primitive generator automatically generates simple GUIs for web

applications that may be used as initial templates.

The voice-to-text module will also assist the community in making the system more convenient to use as it

accepts voice commands.

These results showed that the recommended AI developer assistant can improve programmer efficiency and

significantly reduces the effort in finishing frequent developer jobs.

6. Conclusion

In this paper, we proposed an online AI assistant tool for developers. This proposed online AI assistant tool is

realized as a set of 6 modules for aiding developers to write and understand code, which contains: a natural

language code generator; an automated debug tool; a research paper summarization module; an user interface

generator; a voice-based programming input method.

Modular structure of the whole system makes system is flexible and extensible at the same time and provides a

systematic development process. The experimental results indicate that

AI-assisted programming aid can effectively increase developers' productivity and learning supports for novices.

There is opportunity for further work in offering support to additional languages, additional development

environments or using more advanced AI models.

7. References

[1] M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, "A Survey of Machine Learning for Big Code and

Naturalness," ACM Computing Surveys, vol. 51, no. 4, pp. 1–37, 2018.

[2] S. Gulwani, O. Polozov, and R. Singh, "Program Synthesis," Foundations and Trends in Programming

Languages, vol. 4, no. 1–2, pp. 1–119, 2017.

[3] M. Chen et al., “Evaluating Large Language Models Trained on Code,” arXiv preprint arXiv:2107.03374,

2021.

[4] Y. Feng, R. Martins, O. Bastani, and I. Dillig, “Program Synthesis using Natural Language,” Proceedings

of the ACM SIGPLAN Conference, 2018.

[5] X. Wang, Y. Liu, and H. Li, “Automated Code Generation using Deep Learning Models,” IEEE

Transactions on Software Engineering, 2020.[6] T. Chen and Y. Zhou, “AI-Based Debugging Techniques for Modern Programming Environments,”

Journal of Software Engineering Research, 2019.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers

for Language Understanding,” Proceedings of NAACL-HLT, 2019.