International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.019 ISSN: 2581-4621

IntelliDev: AI-Powered Command Line Tool for Developers

Dr. B. Rama Mohan[1], Abhed Chainani[2], A. Krishna Kartheek[3], Ch. A. S. Praneeth[4], Kaustubh Shakelli[5]
[1]Professor, Dept. of CSE, Hyderabad Institute of Technology and Management, Telangana
[2], [3], [4], [5]UG Students, Dept. of CSE, Hyderabad Institute of Technology and Management, Telangana

Abstract— **I**ntelliDev is a cutting-edge terminal tool designed specifically for developers, seamlessly integrating advanced AI capabilities by harnessing the power of Language Models (LLMs). With a strong focus on user accessibility, it features a user-friendly interface with predefined prompts tailored for various AI-assisted tasks crucial to development. These tasks include conducting thorough code reviews, automatically generating commit messages, and creating tests to enhance code quality. IntelliDev promotes proactive development practices by simplifying coding workflows without requiring extensive technical knowledge. It offers a streamlined and intuitive experience, allowing developers to efficiently leverage AI while maintaining the confidentiality and integrity of the development process.

Keywords— **AI-powered tool, code reviews, command line interface, developer productivity, language models (LLMs), software development.**

INTRODUCTION

In the realm of software development and interaction with computer systems, the Command Line Interface (CLI) serves as a powerful and essential tool. Unlike graphical user interfaces (GUIs) that rely on visual elements such as icons and windows, the CLI embraces a text-based approach, presenting users with a command prompt to input textual commands for performing various tasks. This method of interaction provides a level of precision and efficiency highly favoured by developers and power users. The CLI plays a pivotal role in multiple facets of software development, including efficient code compilation and execution, version control, task automation, and scripting. Developers utilize the CLI to compile source code into executable binaries, manage source code repositories through version control systems, and automate repetitive tasks via build tools and scripting languages. The CLI's capacity for customization and integration with Continuous Integration/Continuous Deployment pipelines further enhances its utility in the development workflow.

Language Models (LLMs) represent a significant advancement in natural language processing (NLP) and artificial intelligence (AI), enabling machines to comprehend, generate, and manipulate human language. These models are trained on vast datasets, allowing them to learn the nuances of language structure, grammar, context, and semantics. By leveraging neural network architectures like transformers and mechanisms such as attention, LLMs can perform a wide array of tasks, from text completion and summarization to language translation and conversational AI.

IntelliDev is conceived as a terminal tool that integrates the capabilities of LLMs to augment and streamline various development tasks. By focusing on user-friendly interfaces and predefined prompts, IntelliDev assists developers in performing code reviews, generating commit messages, and creating tests, all while maintaining a secure and robust environment for AI interactions. This innovative tool aims to simplify coding workflows, enhance productivity, and foster proactive development practices without requiring developers to delve deeply into the intricacies of AI technology.

LITERATURE SURVEY

1. Large Language Models for Software Engineering: A Systematic Literature Review - This research explores how Large Language Models (LLMs) can improve software engineering (SE). It analyses different LLM types, data preparation for LLMs in SE, and techniques to optimize and evaluate them. Finally, it looks at real-world applications of LLMs in SE tasks. Overall, the paper explores the potential of LLMs to revolutionize SE. [Ref 1]
2. A Survey on Machine Learning Techniques for Source Code Analysis - This study explores how machine learning (ML) and deep learning (DL) are used in software engineering for tasks like code completion and analysis. It analyses 479 studies (2011-2021) on 12 code analysis tasks. The findings show a rise in ML/DL usage and identify common steps like model creation and data training used in these tasks. [Ref 2]
3. Software Testing with Large Language Models: Survey, Landscape, and Vision - This paper reviews how large language models (LLMs) are used in software testing. It analyses 102 studies and explores how LLMs are used for different testing tasks, the types of LLMs used, and challenges and benefits of this approach. The paper also details the performance of LLMs in generating unit test cases and explores using LLMs to generate system-level test inputs. Overall, the paper explores the potential of LLMs to improve software testing. [Ref 3]
4. Language Models for Code Completion: A Practical Evaluation – In this paper researchers evaluated the real-world effectiveness of three code completion models (InCoder, UniXcoder, CodeGP) by analysing a massive dataset (2 million completions) from over 1200 programmers across 12 programming languages. Using various metrics (acceptance rate, similarity scores), the study offers a comprehensive picture of how well these models perform in real-world coding, providing insights into their strengths, weaknesses, and effectiveness for different languages. [Ref 4]

EXISTING SYSTEMS

1. Warp - Warp is an AI-powered terminal emulator that enhances the command line experience. Its AI assistant suggests commands, fixes errors, and translates natural language. Warp also offers an IDE-like editing experience with multi-cursors, selection, and familiar key bindings. It stores command history and allows creating reusable workflows ("blocks") that can be shared for collaboration and streamlining development
2. Github Copilot - GitHub Copilot is an AI code completion tool that speeds up coding by suggesting code in real-time. It understands context, works across languages, and can even generate code based on descriptions. It also helps with error correction and provides documentation, all while learning from developers to improve its suggestions.
3. DhiWise - DhiWise streamlines the mobile and web app development process by automatically transforming design mock-ups into clean, well-structured code. This eliminates the need for developers to write everything by hand, boosting their productivity. DhiWise achieves this through a combination of functionalities: automation of repetitive tasks, generation of reusable code components, and an integrated AI assistant named WiseGPT. WiseGPT personalizes code suggestions based on the existing codebase, further ensuring code quality and efficiency. This comprehensive approach saves developer’s valuable time while simultaneously improving the overall quality of the code being produced.

PROBLEM STATEMENT

AI's potential to transform software development is held back by complex, developer-unfriendly interfaces. This hinders the adoption of AI tools for tasks like code reviews and security, despite their potential benefits. Moreover, security concerns arise when using AI with sensitive code. A user-friendly tool that simplifies AI integration while prioritizing security and confidentiality is crucial to bridge the gap between developers and AI, ultimately fostering a more accessible and secure future for AI-powered development.

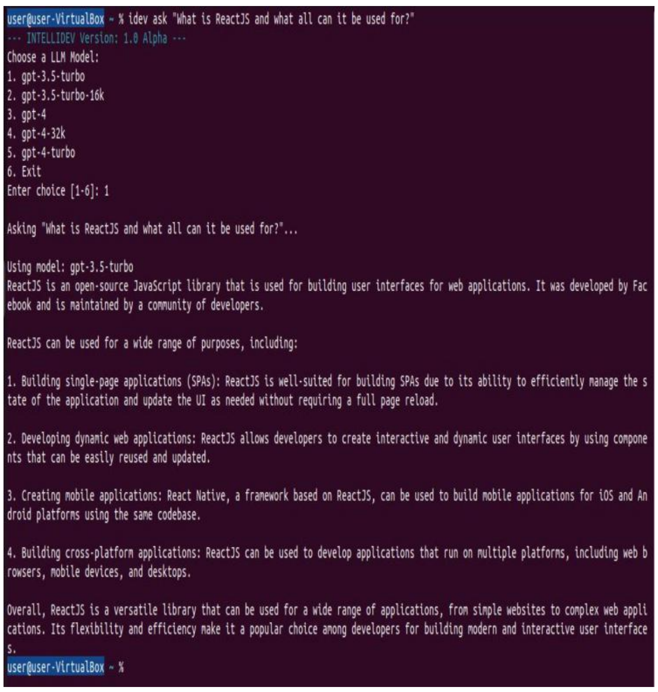
PROPOSED SYSTEM

For IntelliDev, creating a user-friendly and accessible AI development tool was paramount. That's why we made two key choices: a familiar interface (CLI) and powerful Large Language Model (LLM) integration.

- The CLI Advantage: Developers spend a significant amount of time within the terminal environment. By opting for a CLI tool, IntelliDev seamlessly integrates into existing workflows without requiring developers to learn a new interface. This familiarity translates to immediate comfort and efficiency. Additionally, the CLI allows for quick interaction and execution of commands, minimizing disruption to the development process. Most importantly, the CLI fosters a secure environment, as sensitive code remains within the user's terminal, mitigating security concerns.
- The Power of LLMs: Large Language Models represent the cutting edge of AI technology for understanding and responding to human language. By integrating the power of LLMs into IntelliDev, we unlock a vast array of functionalities that directly benefit developers. LLMs can handle diverse tasks, including generating code suggestions, creating test cases, and even offering code review insights. These capabilities streamline development processes and free developers to focus on more complex problem-solving. Furthermore, advancements in LLM technology are rapid. This ensures that IntelliDev will continue to improve and offer even more powerful features as LLM capabilities evolve.

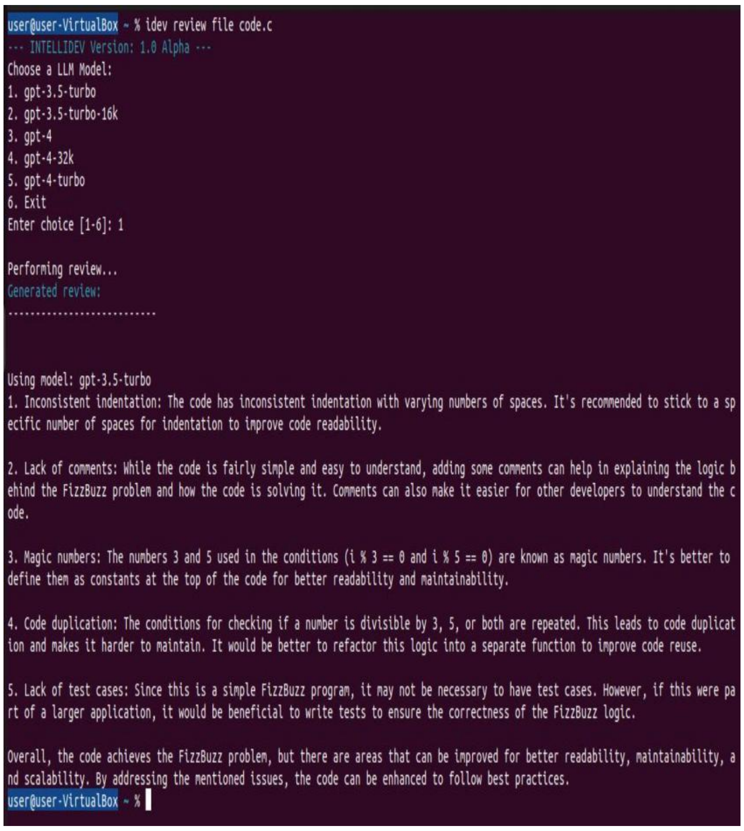
In essence, the combination of a familiar CLI and advanced LLM integration empowers IntelliDev to provide an accessible and effective AI assistant for developers. This approach prioritizes user-friendliness, security, and core functionalities, ultimately aiming to streamline workflows and empower developers to leverage the true potential of AI within their coding endeavours.

RESULTS



```
user@user-VirtualBox ~ % idev ask "What is ReactJS and what all can it be used for?"
--- INTELLIDEV Version: 1.0 Alpha ---
Choose a LLM Model:
1. gpt-3.5-turbo
2. gpt-3.5-turbo-16k
3. gpt-4
4. gpt-4-32k
5. gpt-4-turbo
6. Exit
Enter choice [1-6]: 1

Asking "What is ReactJS and what all can it be used for?"...

Using model: gpt-3.5-turbo
ReactJS is an open-source JavaScript library that is used for building user interfaces for web applications. It was developed by Fac
ebook and is maintained by a community of developers.

ReactJS can be used for a wide range of purposes, including:

1. Building single-page applications (SPAs): ReactJS is well-suited for building SPAs due to its ability to efficiently manage the s
tate of the application and update the UI as needed without requiring a full page reload.

2. Developing dynamic web applications: ReactJS allows developers to create interactive and dynamic user interfaces by using compone
nts that can be easily reused and updated.

3. Creating mobile applications: React Native, a framework based on ReactJS, can be used to build mobile applications for iOS and An
droid platforms using the same codebase.

4. Building cross-platform applications: ReactJS can be used to develop applications that run on multiple platforms, including web b
rowsers, mobile devices, and desktops.

Overall, ReactJS is a versatile library that can be used for a wide range of applications, from simple websites to complex web appli
cations. Its flexibility and efficiency make it a popular choice among developers for building modern and interactive user interface
s.
user@user-VirtualBox ~ %
```

As shown in the above figure, IntelliDev simplifies AI integration for developers. Users can directly submit prompts in the terminal and receive LLM-powered responses within the same interface.



```
user@user-VirtualBox ~ % idev review file code.c
--- INTELLIDEV Version: 1.0 Alpha ---
Choose a LLM Model:
1. gpt-3.5-turbo
2. gpt-3.5-turbo-16k
3. gpt-4
4. gpt-4-32k
5. gpt-4-turbo
6. Exit
Enter choice [1-6]: 1

Performing review...
Generated review:
...........................



Using model: gpt-3.5-turbo
1. Inconsistent indentation: The code has inconsistent indentation with varying numbers of spaces. It's recommended to stick to a sp
ecific number of spaces for indentation to improve code readability.

2. Lack of comments: While the code is fairly simple and easy to understand, adding some comments can help in explaining the logic b
ehind the FizzBuzz problem and how the code is solving it. Comments can also make it easier for other developers to understand the c
ode.

3. Magic numbers: The numbers 3 and 5 used in the conditions (i % 3 == 0 and i % 5 == 0) are known as magic numbers. It's better to
define them as constants at the top of the code for better readability and maintainability.

4. Code duplication: The conditions for checking if a number is divisible by 3, 5, or both are repeated. This leads to code duplicat
ion and makes it harder to maintain. It would be better to refactor this logic into a separate function to improve code reuse.

5. Lack of test cases: Since this is a simple FizzBuzz program, it may not be necessary to have test cases. However, if this were pa
rt of a larger application, it would be beneficial to write tests to ensure the correctness of the FizzBuzz logic.

Overall, the code achieves the FizzBuzz problem, but there are areas that can be improved for better readability, maintainability, a
nd scalability. By addressing the mentioned issues, the code can be enhanced to follow best practices.
user@user-VirtualBox ~ %
```

The figure showcases IntelliDev's code review powered by an LLM. The model analyses your code to identify areas for performance enhancement, ensure adherence to coding standards, and recommend refactoring techniques.

CONCLUSION

IntelliDev incorporates advanced Large Language Models (LLMs) into familiar terminal workflows, enhancing software development processes. This AI-driven assistant evaluates code for quality and performance and integrates smoothly with existing terminal tools. Its modular architecture and customizable workflows support a variety of projects, thereby helping developers create robust software solutions more efficiently.

References

[1] HOU, X. et al. (2024). Large Language Models for Software Engineering: A Systematic Literature Review. https://arxiv.org/pdf/2308.10620.
[2] SHARMA, T. et al. (2022) A Survey on Machine Learning Techniques for Source Code Analysis. https://arxiv.org/pdf/2110.09610
[3] Junjie, W. et al. (2024) Software Testing with Large Language Models: Survey, Landscape, and Vision. https://arxiv.org/pdf/2307.07221
[4] Izadi, M. et al. (2024) Language Models for Code Completion: A Practical Evaluation. https://arxiv.org/ftp/arxiv/papers/2402/2402.16197.pdf
[5] Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020, September 18). Codebert: A pre-trained model for programming and natural languages. https://arxiv.org/abs/2002.08155
[6] Naser, S. et al. (2020) Rate-splitting multiple access: Unifying noma and SDMA in miso VLC channels. https://arxiv.org/abs/2007.13560
[7] Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Scott Yih, Luke Zettlemoyer, and Mike Lewis. Incoder: A generative model for code infilling and synthesis. In The Eleventh International Conference on Learning Representations, 2023
[8] Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A Transformer-based Approach for Source Code Summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 4998–5007. https://doi.org/10.18653/v1/2020.acl-main.449