



UNIVERSITY OF LEEDS

This is a repository copy of *Automatic Code Commenting in Integrated Development Environments Based on Indirect Interaction with Chatbots*.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/202981/>

Version: Accepted Version

Proceedings Paper:

Fabiyi, S. orcid.org/0000-0001-9571-2964 and Ajibuwa, O. (2023) Automatic Code Commenting in Integrated Development Environments Based on Indirect Interaction with Chatbots. In: 2023 International Scientific Conference on Computer Science (COMSCI). 2023 11-th International Scientific Conference COMPUTER SCIENCE, 18-20 Sep 2023, Hotel Lazur, Sozopol, Bulgaria. IEEE . ISBN 979-8-3503-2526-3

<https://doi.org/10.1109/COMSCI59259.2023.10315818>

This is an author produced version of a conference paper accepted to the 2023 11-th International Scientific Conference COMPUTER SCIENCE, made available under the terms of the Creative Commons Attribution License (CC-BY), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Reuse

This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:

<https://creativecommons.org/licenses/>

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.



eprints@whiterose.ac.uk
<https://eprints.whiterose.ac.uk/>

Automatic Code Commenting in Integrated Development Environments Based on Indirect Interaction with Chatbots

Samson Damilola Fabiyi* and Opeyemi Ajibuwa[§]

*School of Computing, University of Leeds, Leeds, United Kingdom, s.d.fabiyi@leeds.ac.uk

[§]School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, USA, ajibuwao@oregonstate.edu

Abstract – Efficient code commenting is critical to improving code readability, maintainability, and collaboration. This paper introduces automated code commenting within an integrated development environment IDE using chatbots. The introduced system, implemented in Python with Selenium, automates the comments generation process, allowing coders to focus on code logic. Results obtained demonstrates successful interactions with chatbots, comment retrieval and handling delay. Challenges identified include instruction selection and extended conversations, offering opportunities for improvement. Future prospects include reusable libraries, user-friendly interfaces, and streamlined code-commenting.

Keywords – automation, chatbot, comment, code, selenium

I. INTRODUCTION

In the rapidly evolving field of software development, efficient code commenting plays a vital role in improving code readability, maintainability, and collaboration among coders [1]. Conventionally, code commenting was a manual and time-consuming process, which requires coders to carefully mark lines of code with explanations. However, with the advent of artificial intelligence (AI) and chatbot technology, a new automated approach to code commenting is surfacing, offering the opportunity to streamline the process and improve overall productivity.

This paper aims at exploring the concept of automatic code commenting within an integrated development environment (IDE) using indirect interaction with chatbots. The proposed method uses AI-powered chatbots to generate comments on lines of code without requiring direct interaction from coders. By automating the code commenting process, coders can focus more on the logic and functionality of their code, while chatbots help provide the necessary contextual explanations and meaningful comments.

The rest of the paper is arranged as follows. Section II introduces the materials and methods utilized in this work. Section III presents the results and discussion derived from evaluating the performance of the proposed system. Section IV concludes the paper and present ideas for future work.

II. MATERIALS AND METHODOLOGY

MATERIALS

A. Integrated Development Environment (IDE)

An Integrated Development Environment [2] is a software package that provides extensive tools and functions for software development. It works as a centralized platform where developers and programmers can write, modify, compile, debug and deploy their programs. Examples of IDEs are Visual Studio, Eclipse, Spyder, etc. In this work, Spyder, an open source IDE designed specifically for scientific computing, data analysis and numerical programming was chosen to evaluate the effectiveness of the proposed approach on codes written in Python programming language.

B. Selenium library

The Selenium library [3], an open source software framework, is well-known for web browsers automation. It incorporates a programming interface which facilitates developers' and programmers' interaction with web applications and automation of web browser processes. Selenium supports multiple programming languages, such as Python, Java, C#, Ruby, and JavaScript, which renders it a versatile and widely used library in various development environments.

C. ChatAI

ChatAI [4] is an AI-powered assistant designed to support users with various tasks and requests. It can interact with users through messaging apps, voice commands, or other interfaces to provide information, schedule appointments, complete tasks, and more. ChatAI is essentially a chatbot - a computer program designed to mimic a conversation with people through text messages or voice communication. ChatAI can respond to users' queries, supply information and carryout tasks using natural language processing and machine learning.

D. Regular expression matching

A regular expression [5], a string that specifies a search pattern, normally consists of regular characters (e.g. letters and numbers) that match each other and special characters with specific meaning in the context of the regular expression. Regular expression matching thus refers to the process of finding and matching patterns in text using regular expressions (usually abbreviated as regex). Using regular expressions, patterns in a text or string can be effectively and flexibly illustrated and recognized. There are online converters which can be used to generate regular

expression for texts. An example of this is Regex Generator [6].

METHODOLOGY

A. Concept

The proposed method involves developing an auto code commenting program within an IDE. The program interacts with a chatbot to automatically generate comments for lines of code without any direct coder interaction.

B. Implementation

The system is implemented using Python programming language and the Selenium library. Selenium allows automated interaction with the chatbot. The implementation involves the following steps:

- **Accessing the Chatbot:** The program uses a function in the Selenium library to access the homepage of the chatbot (Chat) via a web browser specified in the program. User login details (username and password) are provided within the program for authentication.
- **Interaction with the Chatbot:** Initially, the chatbot displays the message, "Hello! How can I help you today?" in its first element. Functions in the Selenium library are then utilized to present our message (instruction) to the chatbot via its next (second) element. The message is presented in a question form which will solicit a suitable response from the chatbot.
- **Retrieving Chatbot Response:** Another function in Selenium is used to select the element containing the chatbot's response and extract the returned response.
- **Formatting the Response:** Since the returned information (response) may include expected comments and additional explanation from the chatbot, a regular expression match operation is performed to extract only the required comments and any related explanation. The implemented program uses the following regular expression: `r"#(.*)"`.
- **Handling Delays:** Due to the dynamic nature of conversations with the chatbot and potential delays in displaying the returned results, a function called `WebDriverWait()` is employed. This function allows the system to wait for a specified duration until the element becomes visible. If the element remains invisible after the specified wait time, an error message is returned.

III. RESULTS AND ANALYSIS

In this section, we present the results obtained from evaluating the program's performance and analyse the findings. Additionally, we explore the challenges encountered during the process and discuss potential opportunities for future improvements and expansions.

A. Results

To evaluate the program, we utilized the following data for various selenium functions:

- i. Username - xxxx@gmail.com (a part of the email address used is hidden for privacy).
- ii. Password - xxxxxxx (again, the password used is hidden for privacy).
- iii. Question supplied to the chatbot: "add a comment to this python programming code: `a = c * d`".
- iv. Waiting duration: 20 seconds.
- v. Condition: `visibility_of_element_located` (it checks to see if the element to be interacted with is present).
- vi. Web browser: Chrome.

Figure 1 depicts the results obtained from running the program, while Figure 2 showcases the output generated by the program in Spyder. As illustrated in Figure 1, the program successfully executed the following steps: opening the home page, locating the login button, using the provided login details to gain access to the chatbot, and supplying the question to initiate a response. Furthermore, Figure 2 demonstrates the effectiveness of the regular expression matching process, which accurately formatted the chatbot's response to include only the relevant comment.

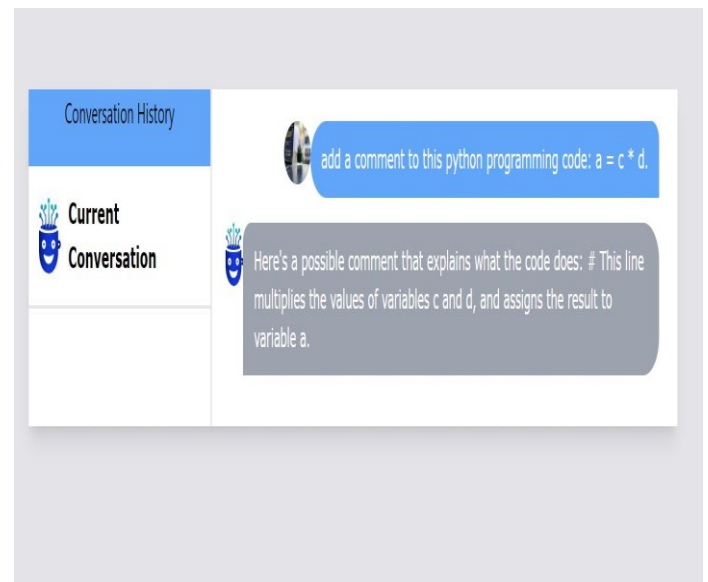
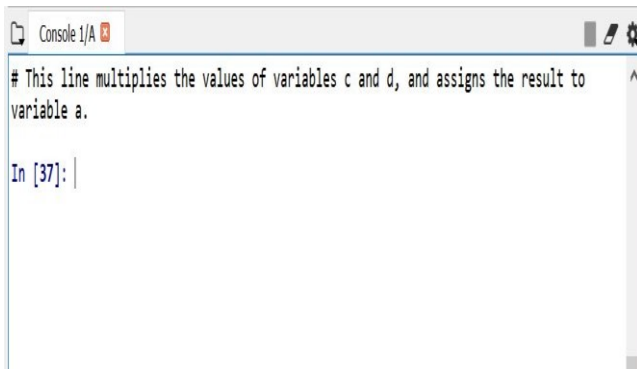


Fig. 1. Conversation with the chatbot

B. Challenges

Throughout the testing phase, several challenges were encountered:

- i. **Finding the right instructions:** The process of identifying and implementing the correct instructions for the chatbot to get the desired comments posed a significant challenge.



```
Console 1/A
# This line multiplies the values of variables c and d, and assigns the result to
variable a.

In [37]: |
```

Fig. 2. Final comment extracted using the regular expression matching process

- ii. Bypassing sites with human/robot checking stage: Certain websites presented additional stages for human or robot verification, making it difficult to bypass these checks.
- iii. Handling extended conversations: The program's current implementation fixed the selection of the third element in the chat as the returned result. However, this approach may be insufficient when dealing with extended conversations. Thus, the code should be updated to dynamically select the most recent message from the chatbot.

C. Opportunities

Despite these challenges, the successful testing of the program opens several promising opportunities for future work. The current work paves the way for the following opportunities:

- i. Development of a reusable library: Creating a library that can be easily called to update and maintain the code consistently.
- ii. Creation of a separate user interface: Designing a dedicated user interface that abstracts away the behind-the-scenes actions, simplifying the process of generating the final code with comments.
- iii. Expanding language compatibility: Extending the scope of the program beyond Python to encompass codes written in other programming languages.
- iv. Streamlining the code commenting process: In order to expedite the code commenting process and promote efficiency, it would be beneficial to include a feature that allows for the opening of a file containing all the code in the IDE. This would then enable the chatbot based system to access and add comments to the entire code in a single go, facilitating the creation of comprehensive and well commented programs. By implementing this feature, the resulting output would consist of a fully commented code, eliminating the need for line-by-line implementation. This streamlined approach would save considerable time and effort, allowing the proposed system to focus on adding meaningful comments to the code as a whole.

IV. CONCLUSION

The proposed automatic code commenting system has been evaluated and the results obtained demonstrates its effectiveness in using chatbot technology to generate contextual explanation and meaningful comments for lines of code within an IDE. Although this work faces some challenges, it also points to promising future prospects, such as the development of a reusable library, a user-friendly interface, reduced execution time, and expanded language compatibility. This work opens the door to more efficient and comprehensive code commenting practices, which will ultimately benefit programmers and advance the field of software development.

V. ACKNOWLEDGEMENTS

For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

REFERENCES

- [1] L. Pascarella, "Classifying code comments in java mobile applications," in Proceedings of the 5th International Conference on Mobile Software Engineering and Systems, 2018, pp. 39–40.
- [2] W. Snipes, E. Murphy-Hill, T. Fritz, M. Vakilian, K. Damevski, A. R. Nair, and D. Shepherd, "A practical guide to analyzing ide usage data," in The Art and Science of Analyzing Software Data. Elsevier, 2015, pp. 85–138.
- [3] S. Raghavendra, Python Testing with Selenium: Learn to Implement Different Testing Techniques Using the Selenium WebDriver. Springer, 2021.
- [4] ChatAI. ChatAI: AI Chat Companion [Online]. Available: <https://chatai.com>.
- [5] D. D. A. Bui and Q. Zeng-Treitler, "Learning regular expressions for clinical text classification," Journal of the American Medical Informatics Association, vol. 21, no. 5, pp. 850–857, 2014.
- [6] Neumann, O. Regular Expression Generator [Online]. Available: <https://regex-generator.olafneumann.org/>

APPENDIX

Data Availability Statement: The data that support the findings of this study are described within the article.