Use of ChatGPT for Assessment Feedback on a Complex Programming Assessment.

Aidan McGowan
Queen's University Belfast
Belfast, N. Irealnd,
aidan.mcgowan@qub.ac.uk

Neil Anderson
Queen's University Belfast
Belfast, N. Irealnd
n.anderson@qub.ac.uk

Chris Smith
Queen's University Belfast
Belfast, N. Irealnd
c.k.smith@qub.ac.uk

***Abstract*— This innovative practice paper explores the integration of ChatGPT into the feedback process for university programming courses, addressing the growing demand for efficient and consistent grading and feedback. The research involved a cohort of postgraduate students enrolled in a compulsory Java programming module. ChatGPT was utilized to generate feedback, which was then compared with traditional feedback methods. The results demonstrated that ChatGPT could produce quick, accurate, and human-like responses, significantly reducing the workload for faculty. Feedback generation was generally well-narrated and detailed, although occasional technical errors required manual adjustments. A survey conducted with the student cohort revealed that 75% of respondents preferred ChatGPT-generated feedback over traditional methods, appreciating its detailed and constructive nature. However, some students expressed a need for more specific guidance, such as code snippets. Lecturers also noted that while the AI feedback was efficient, it sometimes lacked the empathetic tone of human-generated comments. Despite these challenges, the overall response from both students and faculty was positive, highlighting the potential of ChatGPT to enhance the feedback process in programming education.**

Keywords—programming, Gen AI, ChatGPT, university, feedback

I. Introduction

As enrolment in university programming courses grows [1], so does the amount of grading and feedback on assessments. Grading and feedback, though often used interchangeably, serve distinct roles in the educational process. Grading is the process of assigning a value to a student's work, typically in the form of a letter grade, percentage, or numerical score, to quantify their performance against predefined criteria or standards [2]. Feedback, on the other hand, involves providing qualitative information aimed at guiding and improving the student's learning process, highlighting strengths, areas for improvement, and offering specific suggestions for development [3]. The process for grading and feedback for programming assessments is time-consuming and labor-intensive [4][2], as each line of code must be meticulously examined and evaluated to ensure accurate marking. Consequently, the faculty staff responsible for grading and feedback face significant pressure and increased workload [5]. Despite these challenges, students greatly value prompt and detailed feedback [4], as it is crucial for their learning and helps them understand their strengths and areas for improvement.

Many universities employ multiple examiners to expedite the marking and feedback process. For this approach to be effective, it requires a robust marking scheme and a subjective grading rubric. If not carefully managed, using multiple markers can introduce its own challenges, often resulting in inconsistencies in grading and feedback. Even when grading consistency is achieved, generating detailed feedback can still pose issues when produced by multiple markers [6]. In some cases, examiners may not provide detailed or constructive feedback. They often have their own interpretation of the grading rubric, leading to inconsistencies in the feedback given. Consequently, students may receive different feedback for similar work, causing confusion and a sense of unfairness. Additionally, examiners' varying levels of expertise and experience can result in differing standards of feedback. Unconscious biases, personal preferences, and individual writing styles may also influence examiners' feedback, leading to unfair or inconsistent evaluations. This can limit the feedback's effectiveness in helping students understand their strengths and areas for improvement, undermining the credibility and objectivity of the assessment process.

Potential solutions to these issues include the development of automated tools for grading and generating feedback for programming modules [7], albeit with varying degrees of effectiveness [8]. Typically, these tools provide grades and feedback to the students based on static analysis techniques, comparing a submission with a reference solution or with a set of correct student submissions. While the automated grading process may often be prompt the feedback is often very limited and dehumanized [8]. For source code the automated feedback is often limited to whether the unit tests have passed or failed, the expected versus the actual output, or how they differ from a provided reference solution. Furthermore, few tools assess the maintainability, readability, or documentation of the source code, with most using static analysis techniques, such as code quality metrics, in conjunction with grading correctness [9]. The scope of use of these largely proprietary tools tends to mean they remain underutilized outside their originating institutions. This reluctance primarily stems from challenges associated with adapting and tracking their usage. Instructors commonly struggle with identifying, understanding, and incorporating diverse systems into their courses, particularly when these systems' formats and approaches do not align neatly with their teaching objectives [10].

The utilisation of Artificial Intelligence in academic teaching and learning is a prominent trending topic of significant interest and concern within most education fields. ChatGPT is a generative AI tool language model developed by OpenAI that has gained unprecedented adoption and widespread use. By leveraging natural language processing and artificial intelligence, ChatGPT provides personalised, interactive, and accessible tools to potentially enhance education. Nevertheless, ChatGPT presents distinct challenges to traditional education, including the potential for online exam cheating, human-like text generation, reduced

critical thinking abilities, and challenges in assessing information generated by the platform [11]. Programming education shares all these issues; moreover, ChatGPT has the potential to be severely disruptive based on its ability to interact with people without programming knowledge to easily solve programming problems ([12],[13]).

The recent explosion in the use of ChatGPT in programming processes in the software industry [14] has underscored its substantial potential within the programming domain [15]. There are several advantages, limitations, and threats as detailed by [16], that the use of ChatGPT brings to programming education. The main advantages include using Natural Language: to enable people with no programming knowledge to solve programming problems. Quick Response: ChatGPT provides immediate feedback to users by providing quick response. Personalised Learning: ChatGPT can provide users with a customised learning experience. Clear Explanations: ChatGPT provides students with clear explanations about programming topics. Debugging and Feedback: ChatGPT helps students identify and fix programming errors. There are also limitations and threats to the use of ChatGPT in learning to program. Unstructured Learning: ChatGPT teaches students about a particular programming language or topic in an unstructured way. Sometimes it is just wrong!

Moreover the ability of ChatGPT to interpret code and present an evaluated natural language based response in conjunction with on a marking rubric presents a significant opportunity to help ease the time, effort and accuracy issues of marking and feedback as identified by Vimalaraj [4] for large scale programming assessments. This research explores a practical application of ChatGPT in delivering consistent, prompt, and valued feedback to programming students at large scale. It examines the viability of using ChatGPT for this purpose, its impact on students, and gathers feedback from lecturers and faculty involved in the development of the system. It explains the development of the system including guidance on prompt engineering and the fine tuning of the prompts. Feedback from an external non-computing student focus group was collected to gauge their experience with AI-generated feedback and helped shape the process and extends the relevance of the study beyond programming and engineering disciplines. The programming students in the study provided with ChatGPT generated feedback on a live assessment. They were subsequently surveyed and quantitative options gathered with qualitative responses also reported.

II. METHODOLOGY

The study involved a cohort of postgraduate students (n=108) taking a compulsory Java programming module over two semesters. The normal marking process for the end-of-course assessment, weighted at 60% was carried out as per tradition and practice in the module however in parallel the course lecturers provided an feedback analysis generated via ChatGPT. The students were surveyed on their views on the ChatGPT feedback and asked to compare with the format and standard of previous non Gen-AI feedback in the module. Lecturers' and markers views on the use of ChatGPT were also sought via interviews.

A. Marking

The lecturer developed an initial marking scheme and rubric. An preliminary marking meeting with all the graders was held to detail the marking criteria. A representative sample was chosen and then the marking process involved applying markers to the sample items, followed by calibration. This calibration step included comparing individual assessments, resolving any discrepancies, and adjusting guidelines as necessary. Criteria refinement was undertaken as needed before marking the remaining items using the revised criteria. Markers were expected to detail their grading per section and include some brief notes per section (Figure 1). This was then used to produce an integer mark for the section, totalling to the overall assessment mark.

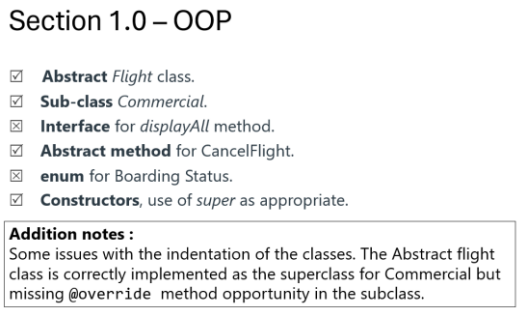

Figure 1 Grading criteria used for section of the assessment

In addition, an overall assessment feedback was authored by each marker to offer a feedback summary for each assessment. The examples showed in Figure 2 and Figure 3 are derived from a subsection of the identical student work, illustrating variations in feedback from distinct markers.

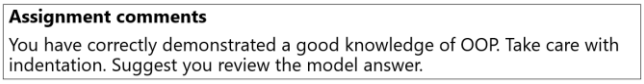

Figure 2 Overall assignment comments (limited verbose)

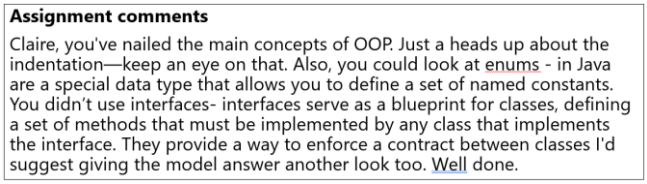

Figure 3 Overall assignment comments (more expansive)

Quality assurance measures were implemented throughout the process, and the results were analyzed for insights and potential improvements. In particular it was noted that there was a variance in the detail and length of assignment comments between the markers. Some markers provided a very brief description as exemplified in Figure 2, while others provided a more narrative description, including detailed technical explanation as shown in Figure 3. Also of note, the length and detail of the feedback was less for those assessments that were marked closer to the deadline for marking completion. While the grading process became more stable and reliable the quality and quantity of narrative feedback remained inconsistent.

B. Developing automated feedback with ChatGPT

When the marking for the students was completed, the agreed marking sheet (Figure 1) for each student was used as a seeded input combined with a prompt (Figure 5) to generate an alternative overall assignment comment using ChatGPT via the authors' *Autofeed App (*Figure 4). Written in Java the *Autofeed App* interacts with the ChatGPT API 3.5 to produce

batches of feedback files for all students. In the context of ChatGPT, prompt engineering involves crafting effective prompts or inputs to guide the AI's responses towards desired outcomes [17]. The process required initial definition of clear objectives for the AI's responses, followed by an iterative process of design and refinement of the prompt [18]. The refinement took into consideration audience tailoring, to ensure clarity and consistency and suitable ethical considerations.

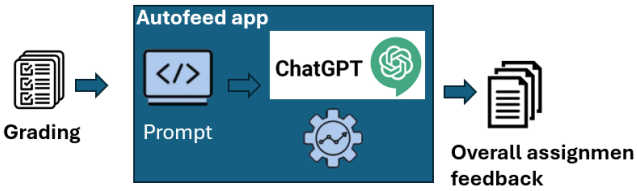

Figure 4 Generation of overall assignment feedback via the Autofeed App

C. *Refining the prompt*

In order to gain a generalised view of the initial AI generated feedback a focus group was convened, comprising students from a non-Engineering background, many of whom lacked programming skills and a deep understanding of technical feedback nuances. They were presented with examples of the previous year's feedback alongside equivalent AI-generated feedback. They overwhelmingly favoured the AI-generated feedback, only realising its origin after it was pointed out. They particularly appreciated the personalisation of the AI-generated feedback. Their comfort with its AI origin stemmed from its alignment with the objective marking criteria. They showed a preference for constructive criticism over generic praise, regarding subjective statements like "*well done*" with scepticism but welcoming comments that encouraged continuous progress, such as "*keep up the hard work*." They placed significant value on specific improvement suggestions and summarising conclusions. Overall, their response was highly positive. Examples of the feedback were also made available to the markers, with the eventual prompt to be used for the live assessment being finalized as shown in Figure 5.

```
PROMPT : Given the following grading for an assignment,produce a feedback for the student
Include the student name in the feedback on the first line.
Include the Other Comments in a separate section.
Tone should be neutral lanaguage.
Spelling should be UK English.
Include a feedback comment for each separate line in the grading criteria
but group into a paragraph for each section.
Relate any areas to improvement to the general lecture materials (slides and labs).
Include a summary conclusion at end.
```

Figure 5 - Refined prompt used to instruct ChatGPT, including direct actions, tone and spelling.

III. RESULTS

The integration of ChatGPT into the feedback process showed promising results. Once refined the prompt produced quick and generally accurate human like responses.

A. *Reviwing the process and feedback examples*

From the lecturers' point of view feedback generation process was efficient, scalable, and provided objective and generally consistent output. Occasional technical errors or inaccuracies were observed in the feedback which necessitated that the lecturers and markers check and if required amend the feedback, but this was still deemed quicker than authoring manually. As illustrated in Figure 6 the feedback was generally well narrated, accurate and detailed.

Assignment comments

Claire, you have demonstrated a good understanding of Object-Oriented Programming principles in your solution. Your use of the Abstract class with a sub-class for Commercial flights is a good organisation choice. Utilising an Interface for the displayAll method would be a good design factor. The use of the abstract method for CancelFlight is a sound design technique and is commendable.

You could improve the solution with the use of enums for Boarding Status. It would be beneficial to review the lecture materials on interfaces and enums to improve these aspects of your solution. Furthermore, the poor indentation in your code makes it difficult to read and should be addressed for better code readability. Overall, your solution shows potential with room for improvement in implementing inheritance and code formatting.

Overall, Claire well done, review the solution for refinements, and keep up the good work.

Figure 6 - Overall assignment feedback from ChatGPT based on the input grading (Figure 1) and Prompt (Figure 5).

The use of a *neutral language* within the ChatGPT prompt was deemed appropriate. In its absence responses included "*awesome work Claire*", "*you have a very strong understanding of OOP*". The latter, while being encouraging is not true in this circumstance based on the marking evidence. The use of "*awesome*" was not an expected tone for UK audiences and would likely led the students to suspect that AI had marked the assessment with limited human interaction.

Without specific prompting ChatGPT was able to relate OOP topics together and provide an overall summative output. "*You have a good understanding of Object-Oriented Programming*", being based on references to core OOP concepts such as Abstract, Inheritance, etc. Another useful output in this theme included "*Furthermore, the poor indentation in your code makes it difficult to read and should be addressed for better code readability*". Again, the linkage between poor indentation and code readability is not specifically called out in the grading but is sound link and of benefit to the student for improvement on their work.

The output from ChatGPT characteristically identified areas that were incomplete i.e. not ticked in the grading criteria and generically linked those areas to the related lecture materials, e.g. "*It would be beneficial to review the lecture materials on interfaces and enums to improve these aspects of your solution*". However, on occasion it erroneously fabricated a reference, e.g. "*see lecture week 4*".

The level of narration was generally considered as appropriate; consider the output for a low scoring assessment Figure 7 compared to higher scoring Figure 8. Both are accurate in tone and action, with the low scoring assessment outlining areas to work on and the high scoring acknowledging achievement but still ending with an encouraging "*keep up the good work*".

Gerard, your Object-Oriented Programming solution lacks several key components.

Firstly, it is important to include an Abstract Flight class and then have a Sub-class Commercial that extends from it. This concept is covered in our lecture slides on inheritance and polymorphism. Additionally, implementing an interface for the displayAll method will help to organise your code better and make it more modular. Furthermore, it is essential to incorporate an Abstract method for CancelFlight and use an enum for Boarding Status to enhance the structure of your program. This is covered in our object-oriented programming labs.
Constructors play a crucial role in initialising objects, so it is advisable to include them in your classes. Also, make sure to use the super keyword appropriately when necessary.

Lastly, work on improving your indentation to make your code more readable and easier to follow.

In conclusion, addressing these areas of improvement will help strengthen your understanding and application of Object-Oriented Programming concepts in your future solutions.

Figure 7 - Feedback from low scoring assessment

Una, OOP solution demonstrates a very strong understanding of abstract classes, subclassing, and interfaces. The Abstract Flight class serves as an effective superclass for the Commercial subclass, showcasing a good use of inheritance. Your implementation of the displayAll method through an interface shows a clear understanding of abstraction.

Overall, you have shown great proficiency in implementing OOP principles in your solution. Keep up the good work!

Figure 8 - feedback from high scoring assessment

Once the prompt was established creating the automated feedback for the entire cohort took less than one minute. However, the output did require manual checking. Around 25-30% of the feedback responses required some amendment, despite having the same prompt and grading format. These included minor format edits to more significant corrections where generated output either incorrectly credited or ignored a marked component or tone or format was deemed incorrect. Feedback responses were sometimes overly formatted, with greetings like "*Dear Ryan*" and sign-offs like "*Regards [name]*". The system will regularly return an error if tab characters, code snippets, or double quotes were included in the prompt. Additionally, the response may skip information related to partially completed work. Moreover, at times it was felt that the responses can on occasion appear harsher and less empathetic than those written by humans. Without these corrections the feedback would inevitably lead to some students questioning not just the feedback but their grade.

B. Feedback from students

A week after their grades and ChatGPT originated feedback were released the whole cohort were surveyed regarding their opinion on the feedback. Overall, 75% of survey respondents (Figure 5) indicated a preference for the ChatGPT feedback over the shorter, list-based, non Gen-AI feedback they received in earlier assessments.

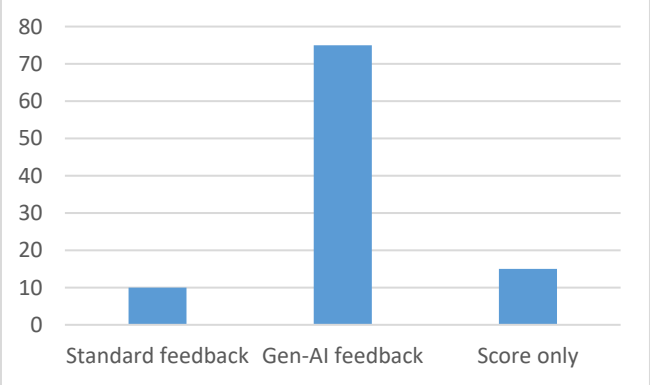

Figure 5 Preferences of the cohort on format of feedback.

Figure 5 illustrates that 80% of the cohort felt that feedback presented a balanced view of their work and in particular 95% (Figure 7) stated that it also signposted how to improve on areas of weakness.

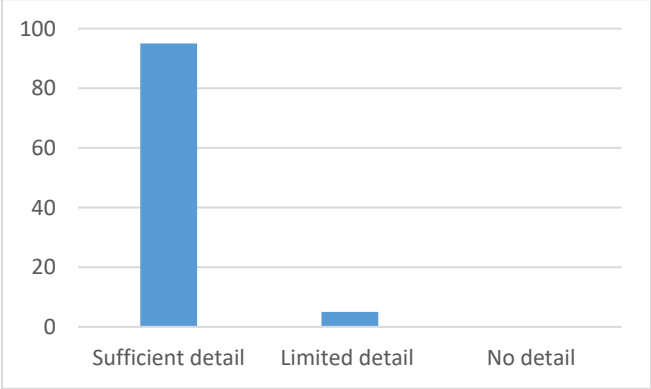

Figure 6 Feedback on detail provided by ChatGPT feedback on weaknesses and how to improve.

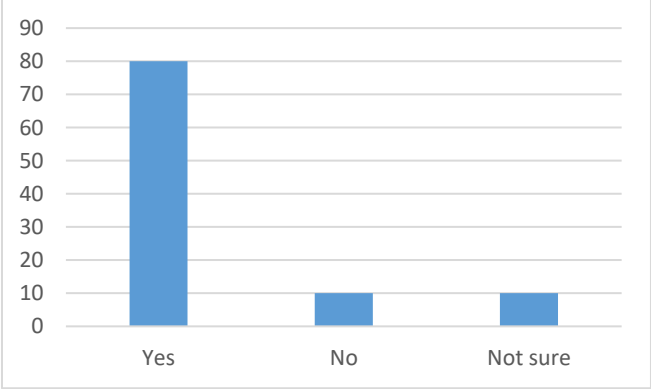

Figure 7 Student responses on whether the Gen-AI feedback addressed strengths and weakness of their submissions

A small number of the students felt that the feedback focused excessively on their mistakes rather than their achievements, creating a disconnect between their grades and the comments. "*The comments seemed to focus on the negative, what I failed to do or did incorrectly. My grade wasn't as negative as the comments*," one student remarked. Despite this, most others found the feedback to be constructive and helpful for future improvement: "*I found the feedback very constructive and extremely useful. It will help me improve my coding in the future*."

A small number also stated that *in places* the comments were quite broad and would be more useful "*if they specified where the improvement could be made*". While the majority of students stated the feedback was of sufficient length many expressed a desire for more detailed specific guidance, such as code snippets, to better understand their mistakes and improve their coding skills. One student noted, "*I'd love to also see code snippets where possible*". As already noted the responses from ChatGPT often ignored code snippets that markers included in their feedback, especially in the Other Notes section (Figure 1). Overall the inclusion of code snippets or other formatting within the prompt with ChatGPT 3.5 was problematic, with code being either ignored or an error being returned.

As part of the feedback process a model answer (video narrated walk through of the generic solution) was also provided. One student explained, "*the feedback is a list of areas for improvement; I find the walkthrough of the ideal solution more beneficial in terms of learning.*" Which perhaps shows the student value for generic model solutions alongside with the provision of a verbose, tailored and detailed feedback. At the other extreme 15% of the students stated they would just prefer their mark without feedback of any format.

Historically after mark release around 5% of the students question or challenge their mark / feedback. In this study only 1% did so. That query related to the score and perception that they had not be accredited for part of their submission and was not directly related to the feedback. Overall, students appreciated detailed feedback, with one stating, "*I found the feedback from the assessment very helpful, and it was nice to receive such detailed feedback*."

IV. DISCUSSION

The integration of ChatGPT into the feedback process for programming assignments showed promising results, producing quick and generally accurate, human-like responses once the prompt was refined. From the lecturers' perspective, the feedback generation process was efficient, scalable, and provided objective and consistent outputs. Although occasional technical errors or inaccuracies required manual review and amendments, this approach was still faster than manually creating feedback. The use of neutral language in the ChatGPT prompt was essential to avoid overly enthusiastic or inaccurate comments, ensuring the feedback was appropriate for the academic context.

The feedback was well-narrated and detailed, effectively linking concepts such as object-oriented programming (OOP) and providing useful recommendations like improving code readability through proper indentation. ChatGPT also identified incomplete areas and linked them to related lecture materials, although it sometimes fabricated references. The feedback for both low-scoring and high-scoring assessments was accurate in tone and content, with the former focusing on areas for improvement and the latter acknowledging achievements while encouraging further efforts. Despite the need for manual checking, the process of generating automated feedback for the entire cohort took less than a minute.

Student feedback was generally positive, with 75% of survey respondents preferring ChatGPT feedback over the shorter, list-based feedback from earlier assessments. Most students found the feedback constructive and helpful for future improvement, appreciating its balance and detailed guidance. However, some students felt the feedback focused too much on their mistakes and expressed a desire for more specific guidance, such as code snippets. The inclusion of a model answer (video-narrated walkthrough) alongside the feedback was found beneficial by some students. Overall, the integration of ChatGPT into the feedback process resulted in fewer student queries or challenges regarding their marks, indicating improved satisfaction with the feedback provided.

A. Limitations of the study

The students involved in the study were post graduate and in the Computing discipline and as such the scope of further study could include undergraduate and other disciplines. The number of students is limited to the size of the cohort and expanding the same study to greater numbers would be of interest as would re-running the same experiment with a new intake cohort.

B. Suggested Areas of Future Work

Improved AI prompting techniques, developing more advanced prompts that better guide the AI in producing accurate and context-appropriate feedback is essential. This includes incorporating mechanisms to avoid fabricated references and ensuring all comments are based on actual assessment criteria. Enhanced error handling is also crucial, involving the implementation of robust error detection and correction mechanisms to minimize manual amendments. It is important to ensure the AI can handle various formatting elements, such as code snippets, tabs, and double quotes, without errors or omissions. Refining the AI's ability to provide empathetic and supportive feedback that aligns with human feedback styles, and training it to maintain a consistent and appropriate tone across all responses, will further enhance its utility. Exploring ways to integrate AI feedback with model answers or detailed walkthroughs can enhance the learning experience, providing a more comprehensive understanding of the material.

Conducting long-term studies to assess the impact of AI-generated feedback on student performance and learning outcomes, and gathering data on how students use and perceive AI feedback over multiple semesters, will identify areas for further improvement.

By addressing these areas, the integration of ChatGPT into educational settings can be further optimized, ultimately enhancing the quality of feedback and supporting student learning in a more effective and scalable manner.

V. CONCLUSION

The integration of ChatGPT into the feedback process for programming assignments in this study has demonstrated significant potential in providing efficient, scalable, and generally accurate feedback. The AI-generated responses, once refined, were able to produce human-like feedback that was well-narrated and detailed, aiding in students' understanding and improvement of their coding skills. Both lecturers and students found the process to be advantageous, with lecturers noting the time efficiency and students appreciating the detailed and constructive feedback.

However, several areas need further refinement. The occasional technical errors and inaccuracies necessitate ongoing human oversight to ensure the quality and reliability of the feedback. Additionally, the AI's tendency to produce overly enthusiastic or inappropriate comments without neutral language prompts, as well as the fabrication of references, highlights the need for more sophisticated prompting techniques. While mostly focused on programming the study does highlight the extensibility to other Engineering disciplines.

REFERENCES

[1] BCS. University Computing departments met with record applicant numbers as AI hits the mainstream. Retrieved May 18, 2024 from https://www.bcs.org/articles-opinion-and-research/university-computing-departments-met-with-record-applicant-numbers-as-ai-hits-the-mainstream/

[2] Brookhart, S. M. (2008). How to give effective feedback to your students. ASCD. Retrieved May 18, 2024 from https://files.ascd.org/staticfiles/ascd/pdf/siteASCD/publications/books/How-to-Give-Effective-Feedback-to-Your-Students-2nd-Edition-sample-chapters.pdf

[3] Hattie, J., & Timperley, H. (2007). The power of feedback. Review of Educational Research, 77(1), 81-112.

[4] H. Vimalaraj et al., "Automated Programming Assignment Marking Tool," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 2022, pp. 1-8, doi: 10.1109/I2CT54291.2022.9824339.

[5] J. Tharmaseelan, K. Manathunga, S. Reyal, D. Kasthurirathna and T. Thurairasa, "Revisit of Automated Marking Techniques for Programming Assignments," 2021 IEEE Global Engineering Education Conference (EDUCON), Vienna, Austria, 2021, pp. 650-657, doi: 10.1109/EDUCON46332.2021.9453889. keywords: {Deep learning;Training;Computer languages;Automation;Computational modeling;Buildings;Training data;Automatic Assessment;Programming;GCNN;Source Code Analysis;Education;Large Class Teaching},

[6] James Williams, David Kane, and Gillian Cappuccini-Ansfield. 2008. Student satisfaction surveys: the value in taking an historical perspective. Quality in Higher Education, 14, 2, 135–155.Chelsea Finn. 2018. Learning to Learn with Gradients. PhD Thesis, EECS Department, University of Berkeley.

[7] Marcus Messer, Neil C. C. Brown, Michael Kölling, and Miaojing Shi. 2023. Automated Grading and Feedback Tools for Programming Education: A Systematic Review. 1, 1 (December 2023), 43 pages. https://doi.org/10.1145/

[8] Cesare Aloisi 2023 The future of standardised assessment: Validity and trust in algorithms for assessment and scoring First published: 17 January 2023 https://doi.org/10.1111/ejed.12542James W. Demmel, Yozo Hida, William Kahan, Xiaoye S. Li, Soni Mukherjee, and Jason Riedy. 2005. Error Bounds from Extra Precise Iterative Refinement. Technical Report No. UCB/CSD-04-1344. University of California, Berkeley.

[9] Hollis-Sando, L., Pugh, C., Franke, K., Zerner, T., Tan, Y., Carneiro, G., van den Hengel, A., Symonds, I., Duggan, P., & Bacchi, S. (2023). Deep learning in the marking of medical student short answer question examinations: Student perceptions and pilot accuracy assessment. Focus on Health Professional Education: A Multi-Professional Journal, 24(1), 38–48. https://search.informit.org/doi/10.3316/informit.997864474732299

[10] D. M. Souza, K. R. Felizardo and E. F. Barbosa, "A Systematic Literature Review of Assessment Tools for Programming Assignments," 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), Dallas, TX, USA, 2016, pp. 147-156, doi: 10.1109/CSEET.2016.48. keywords: {Programming profession;Education;Systematics;Bibliographies;Databases; Data mining;Assessment tools;Programming assignments;Mapping study},

[11] Rahman, M.M.; Watanobe, Y. (2023) ChatGPT for Education and Research: Opportunities, Threats, and Strategies. Appl. Sci. 2023, 13, 5783. https://doi.org/10.3390/app13095783Prokop, Emily. 2018. The Story Behind. Mango Publishing Group. Florida, USA.

[12] Surameery,N & Shakor, M.Use chatgpt to solve programming bugs. International Journal of Information Technology & Computer Engineering (IJITC), 3 (1) (2023), pp. 17-22.

[13] Yilmaz, R., & Karaoglan Yilmaz, F. G. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. Computers in Human Behavior: Artificial Humans, 1(2), 100005. https://doi.org/10.1016/j.chbah.2023.100005

[14] Dwivedi, Y. K., Kshetri, N., Hughes, L., Slade, E. L., Jeyaraj, A., Kar, A. K., Baabdullah, A. M., Koohang, A., Raghavan, V., Ahuja, M., Albanna, H., Albashrawi, M. A., Al-Busaidi, A. S., Balakrishnan, J., Barlette, Y., Basu, S., Bose, I., Brooks, L., Buhalis, D., . . . Wright, R. (2023). Opinion Paper: “So what if ChatGPT wrote it?” Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. International Journal of Information Management, 71, 102642. https://doi.org/10.1016/j.ijinfomgt.2023.102642

[15] Yilmaz, R., & Karaoglan Yilmaz, F. G. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. Computers in Human Behavior: Artificial Humans, 1(2), 100005. https://doi.org/10.1016/j.chbah.2023.100005

[16] Ramazan Yilmaz, Fatma Gizem Karaoglan Yilmaz, Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning, Computers in Human Behavior: Artificial Humans, Volume 1, Issue 2, 2023, 100005, ISSN 2949-8821,https://doi.org/10.1016/j.chbah.2023.100005.

[17] Partha Pratim Ray, ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope, Internet of Things and Cyber-Physical Systems, Volume 3, 2023, Pages 121-154, ISSN 2667-3452, https://doi.org/10.1016/j.iotcps.2023.04.003.

[18] N. Tajbakhsh, J.Y. Shin, S.R. Gurudu, R.T. Hurst, C.B. Kendall, M.B. Gotway, J. Liang Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans. Med. Imag., 35 (5) (2016), pp. 1299-1312