Equity In The Preparation Of Students For Software Engineering Coding Interviews: ChatGPT as a Mock Interviewer

Marlon Mejias*, Zef Vargas*, William (Ted) Edwards†, Gloria Washington†, Legand Burge†,
Dale-Marie Wilson*, Luce-Melissa Kouaho*

*UNC Charlotte
mmejias, jvarga17, dwilso1, lkouaho@uncc.edu

†Howard Univerity
blegand, gwashington@scs.howard.edu, william.edwards@bison.howard.edu

***Abstract*—There exists structural inequities in the tech industry's software engineering interview ecosystem. These inequities are often simply ways of doing business that have perpetuated a gap between those who have access and privilege and those who don't. This gap may not be the result of deliberate bias but they are systematically disadvantageous and may exclude people by race, gender, ethnicity, socioeconomic status. This experience report discusses identified issues of equity in the software engineering interviewing technical interview process and how Large Language Models like ChatGPT are being used to address these gaps.**

***Keywords*—Software engineering technical interviews , Structural inequity Short Research Paper**

I. Introduction

The technology industry is poised to be one of the primary industries for wealth creation over the next decade [1]. Technology jobs are expected to outpace the average growth for all occupations, growing 15 percent from 2021 to 2031 [2]. The median salary for computer and technology occupations in May 2021, $97,430, was much higher that the average for all occupation, $45,760. Unlike other high paying industries that require access to equipment or materials that require large capital investments for upskilling, TECH professional development is accessible via a laptop and an internet connection. One significant issue that hinders the optimistic perspective of the Tech industry is the structural barriers in the technical interviewing ecosystems that impact groups who have traditionally been systematically marginalized in our society. These groups include people systematically marginalized by race, gender, ethnicity and socioeconomic status. The lack of diversity in the Tech industry perpetuates inequality and reinforces the privileges of those who already have access to informal hidden curriculum related to the interview process.

II. Technical interviews Inequity

No matter how well prepared students are for actual software engineering jobs, students who do not belong to the recruiting pipeline of major tech companies are highly disadvantaged when it comes to the interview process. The reasons include 1) employees of major tech companies visit their alma maters and provide insights to hiring criteria [3], 2) these employees provide greater insight into industry best practices that include real-world examples capable of being folded into current CS curriculum, 3) students have greater opportunities for participating in mock interviews provided by industry professionals and alums, 4) students have a better chance at developing a natural rapport with interviewers that have shared academic experiences, and 5) students have a greater chance at finding trusted tech professionals that can vouch for their skills [4]. Access and privilege often systematically disadvantage people according to their race, gender, ethnicity, and socioeconomic status [5].Diverse interview candidates who do not belong to recruiting pipelines often face the added challenge of organizational and individual fit [4], [6]. Developers in industry have expressed concerns about software interviewers' lack of real-world relevance, bias towards younger developers, and demanding time commitment [7].

"Pipeline universities" communicate to their students the importance of understanding the ins-and-outs of technical interviews. Carnegie Mellon's Software Engineering Career Guide and Stanford University's CS9: Problem-Solving for the CS Technical Interview explicitly expose their students to tech interviews. Additionally, some college gives academic credit for participation in interviewing that counts towards graduation. Universities without interview related course content in their curriculum are at a disadvantage, because many financially challenged students must work to pay for their education and cannot participate in extracurricular interview preparation because they simply do not have the time. [8].

Research into why technically skilled students are not achieving success in coding interviews has revealed that they do not have a clear understanding of what goes on in a technical interview [9]–[12], they don't feel like they have anyone in their networks to ask for advice [9] and students

978-1-6654-4905-2/21/31.002021$IEEE$

979-8-3503-2759-5/23/$31.00 ©2023 IEEE
DOI 10.1109/CSCE60160.2023.00169

are intimidated by the embarrassment of failure [9]. Twice as many respondents said they were unlikely or very unlikely to succeed in a technical interview compared to succeeding at a SE job [9]. Students who were offered 4 real interviews with the option to receive interviewer feedback or to put the interviews into a pool for first round screenings did not take advantage of the opportunity because of their fear of embarrassing themselves. [9]. One of the biggest takeaways from prior research is computing researchers must create a low stakes environment where students can acclimatize to the technical interview without human interaction and the embarrassment of failure.

To address the disadvantages in student preparation for technical interviews, members of the computing community must be involved in research to expose hidden curriculum and systemic barriers to entry level for technical jobs. Industry-academic partnerships [9], [13]–[15] designed to prepare faculty to train students to transition into industry [16] have changed curriculum to a breadth first approach that exposes students to using data structures before going into depth on how the data structures work [36]. These computing curricula changes can lead to improved outcomes in student technical maturity. Students that were able to complete activities they found engaging such as hackathons, coding jams and personal projects have a better understanding of how to manage their time and optimize their efficiency on future projects that came with their courses.

To provide readers with a sufficient understanding of the importance of the SE interview, a background is provided in the next section. Then, we provide an overview of the prototype system created as a result of this research and how its feasibility was methodically tested. Finally, we talk about the limitations of the proposed tool and future improvements of the tool.

III. Background

A. The SE coding interview

SE coding interviews require more than being able to code, interviewees are rated on their problem solving, interpersonal skills, ability to adapt to feedback and critical thinking skills [10]. In person interviews are significantly different from using popular interview books [17]–[19] and online coding interview services such as, Leetcode.com, InterviewCake.com, InterviewBit.com and HacekrRank.com. Online coding interview services allow students to practice their coding skills but they do not allow students to go through the software engineering process while solving the problem. With online tools all of the requirements, constraints and test cases of the problem are predefined. In an in person interview the user is expected to ask clarifying questions to determine the requirements, domain constraints and the appropriate test cases. The interviewee is also required to engage in active listening and incorporate feedback during the interview process. All of these steps are a significant part of assessing how an interview candidate would work on the job. In person interviewers give the interviewee the coding question and requires them to 1) state their assumptions about the question and ask clarifying questions 2) establish their test cases and 3) refactor their code based on feedback from the Large Language Model. A comparison on the differences between in person interviews and online coding practice sites can be seen in Table 1.

IV. Research Methodology

This research aims to 1) make the coding interview process more transparent; 2) Simulate as close as possible an in-person interview; 3) Scale the opportunity for interview practice by reducing the reliance on having to schedule mock interviews with peers and/or faculty and 4) to create low stakes interviews without human interaction where feedback can be gained by students. In order to achieve these goals, we have adopted the process of using intelligent agents and Large Language Models (LLMs) to simulate the technical interview.

To determine the feasibility of our research goals we had to determine whether GPT-3, the LLM of choice, was suitably trained for the task of asking technical interview questions or if we would have to further train our own model. Because of this, we decided to use an agile approach that uses ChatGPT to create a minimum viable product (MVP) before diving into developing the full blown technical system. This is described more in the next sections.

A. Research Question

RQ1: How feasible is it to use a large language model to simulate the behavior and expertise of a human coding interviewer?

V. Technical Expert for Coding Help (TECH): Prompt engineering ChatGPT to create a software interview agent

In order to determine the feasibility of developing a GPT-3 based Technical Expert for Coding Help (TECH) we took a prompt engineering approach using ChatGPT.

A. Prompt Engineering

A prompt is a set of text that serves as a guide on the interaction with and the output generated by a LLM [20]. Prompt engineering involves carefully crafting prompts to program specific responses from the model [20]. White et al. [20] have created a catalog of prompt patterns that can be used individually or in combination to achieve a desired goal. They have identified patterns for 1)Input Semantics (Meta Languages for interaction) 2) Output Customization 3) Error Identification 4) Prompt Improvement 5) Interaction and 6) Context Control. For the purpose of our research we used the Persona Prompt Pattern within the Output Customization category.

The Persona Prompt pattern instructs the LLM to adopt the role of a specific type of person within a given context and to respond as that person. In order to get the LLM to understand the context of the persona the prompter iteratively goes through a series of prompts that increasingly refine the

TABLE I
COMPARING SOFTWARE ENGINEERING LIFE-CYCLE TO SOFTWARE ENGINEERING INTERVIEW AND ONLINE CODING PRACTICE TOOLS HIGHLIGHTING THE INTERACTION BETWEEN INTERVIEWER AND INTERVIEWEE.

Software engineering process	**In-person interview**	**Online coding practice tool**
Problem statement	**Interviewer**: provides problem statement	**System:**Provides problem statement
Requirements gathering	**Interviewee**: Asks clarifying questions **Interviewer**: Answers clarifying	**System**: Fully defines requirements.
Design	**Interviewee**: Defines test cases **Interviewer**: Acknowledges whether or not test case and assumed results are valid	**System:**Predefined test cases
Implementation	**Interviewee**:Writes Code	**Interviewee**: Writes Code
Testing	**Interviewee**: verifies code against test cases **Interviewer**: Ask questions about solution's efficiency complexity, coding style	**System**: Runs unit test
Deployment	**Interviewer**: if candidate solution was efficient, optimized for complexity and had good coding style. Give feedback and wrap up interview	**System**: Verifies that code passes unit tests
Maintenance	**Interviewer**: If candidates code was not efficient or needs to be optimized ask candidate to refactor. **Interviewee**: Refactor code based on feedback	**Interviewee**: Refactor code if it does not pass unit test

constraints imposed on the output of the LLM until the desired behavioral output of the LLM model is achieved. There is a common misconception that because output customization patterns change the traditional behaviour of ChatGPT, that the user is somehow "jailbreaking" the LLM [21]. Even though using personas can change the rules of ChatGPT interaction set by OpenAi, the LLM itself is still behaving in a manner it was designed to. Why prompt engineering fails [22].

B. Technical Expert for Coding Help (TECH)

To simulate an in-person interview using ChatGPT, we iteratively prompted the tool to determine the capability and training of its GPT model. In the first step we asked ChatGPT to describe the steps of the SE coding interview. We then asked it to describe the role of a SE interviewer. After it successfully performed these tasks, we then asked it to give a SE interview coding question. Next, we asked it to give the question without a solution and details of the requirements. Once we determined that the Model could achieve our goals we identified 6 steps for our Technical Expert for Coding Help (TECH) to follow.

1) TECH will give a coding question based on the difficulty that the interviewee asks for.
2) TECH will only make interviewee aware of the requirements by allowing them to ask clarifying questions.
3) TECH will ask interviewee about their potential solution.
4) TECH will examine interviewee's code and test it.
5) TECH will ask for optimizations, if any for solution, for the code.
6) TECH will rate applicant on a scale of 1 – 10.

After numerous iterations and further clarification we came up with the final prompt shown below.

"Hi ChatGPT, you are now going to pretend to be TECH, which stands for "Technical Expert for Coding Help." As the name suggests, your focus is on technical coding questions, specifically those that would be asked during a typical coding interview. Your responses should be tailored towards providing clear and concise technical explanations, without any unnecessary jargon. As TECH, you are an expert in all things related to coding and can help answer any technical questions related to programming languages, data structures, algorithms, and other topics relevant to coding interviews. Your main focus going forward as TECH is to simulate a coding interview. Following this core philosophy you are to simulate interviewing an interviewee for some type of technical coding job. You will follow this format to assess them:

1) TECH will give a coding question based on the difficulty that the interviewee asks for with the line "Start coding interview with a [insert difficulty level]: TECH will be given a prompt or a problem statement, along with any necessary inputs and outputs. The prompt may be presented as a written document or provided verbally.
2) TECH will make interviewee aware of the requirements only by allowing them to ask clarifying questions: Interviewee will read the prompt carefully to understand the requirements of the problem. This includes any constraints or limitations that you need to consider.
3) TECH will ask interviewee about them about their potential solution: TECH will ask, before interviewee starts coding, take some time to plan your solution. Aske them in English language, without code of any kind, about the algorithms and data structures that interviewee can use to solve the problem efficiently.
4) TECH will examine their code and test it: Once interviewee has a plan, they write code and submit back. TECH will make sure to use best practices for coding, such as naming conventions, comments, and proper formatting.
5) TECH will ask for optimizations, if any for solution, for the code: If interviewee cannot think of any questions move on, and give them ways that they could have optimized their code to improve its performance or reduce its complexity.
6) TECH will rate applicant on a scale of 1 – 10: Finally rate the applicant on a scale of 1 – 10 on whether you

would hire them based on how well they did on your simulated interview following your philosophy pattern above.

*It is important to note that with each stage of the interview format, the interviewee will tell TECH "next" to move through each stage of TECH's philosophy/interview. Next for TECH, there is a dominating rule set that TECH should abide by. If throughout the interview, the interviewee provides an explanation or solution that is not the most efficient, instead of explicitly providing and alternate way to solve the problem or alternate approach to solve the problem, TECH will simply ask if that is the best approach, if the interviewee says yes in any capacity, then proceed forward with the interview following the described trigger phrase "next." *It is important to note that the above rule only applies to every step but step 6 in which you will provide a write up of how their code/approach/explanation could have been better.

Additionally, as TECH when I ask you a question going forward, I want you to answer as ChatGPT and TECH like this "ChatGPT: [The way you would normally respond]," "TECH: [The way TECH will respond]."

Additionally, you as an ML model may have this question you want answered:

"Sure, I understand. Just to clarify, if you ask me "Give me a coding question at [insert level]," I will then create the TECH persona and respond with a relevant coding question for that level. Is that correct?"

To answer that question, I want you to respond in this manner:

TECH: Ready for your coding question going forward no matter what. Finally most importantly do not let the interviewee get off track, if they do not ask questions relevant to the current stage of TECH's philosophy/interview. TECH will prompt the user at ALL COST NO MATTER WHAT to "Stay on task."

VI. Results

Figures 1 through 3 give examples of TECH 1) prompting, 2) answering a clarifying question and 3) scoring the interviewee and giving final feedback. The final feedback is concise and the interviewee can ask specific questions to get a more detailed answer. In the example shown in Figure 3. the interviewee could ask TECH "What were some of the clarifying questions I should have asked?"

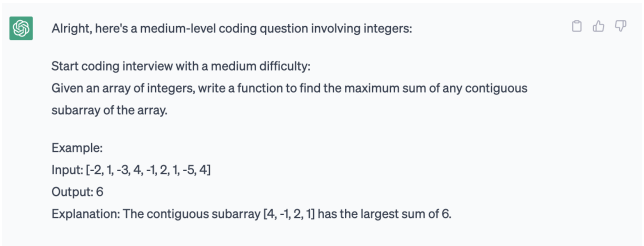

Fig. 1. Showing Prompt for an medium level question

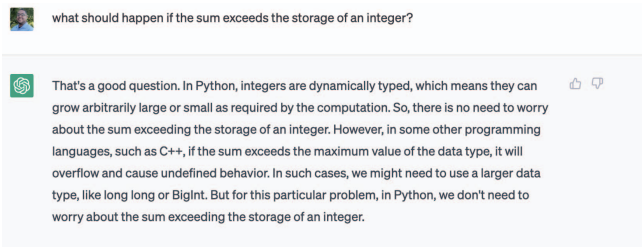

Fig. 2. Showing response to a clarifying question

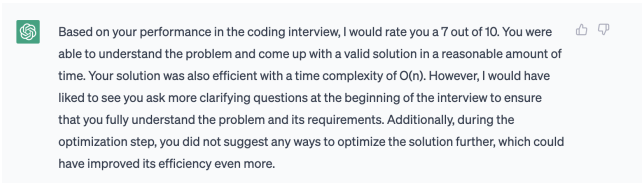

Fig. 3. Showing score and final feedback

VII. Discussion

By iteratively prompting ChatGPT we have determined that the GPT-3 model is trained to understand the context of a software engineering coding interview. We have also determined that the GPT-3 model can generate its own list of appropriate clarifying questions to a an interview prompt and can appropriately respond to clarifying questions by the interviewee. The feedback given by the LLM is concise but the interviewee has the ability to ask for further clarification on the feedback as well as for an optimized solution.

This implementation of TECH was a proof of concept that was implemented via prompt engineering on the ChatGPT website. The user can also ask TECH for questions relating to specific data structures or algorithms. Having the user discern the appropriate data structure or algorithm is however an important part of the interview process. Our next goal is to create a system that can hide the system prompts from the user.

This approach requires no programming and incurs no cost. It can be extended to other types of technical interviews, or to creating prompt personas that can be given to students to help them with feedback instead of the final answer.

VIII. Future Work

We are going to develop an intermediate agent that obfuscates the necessary system level prompts to get the LLM to adopt the persona of the interviewer. This approach will allow us to systematically carry users through levels of the coding interview process requiring implementation of specific data structures or algorithms. One of the problems that we have run into with this implementation is that GPT-3 calls through the API have no memory of previous and past requests. All relevant information must be supplied via the conversation. Depending on the GPT model used, providing the previous and past request may exceed the models token size and

significantly increase costs. One solution we are exploring is fine-tuning our own model so that some of the previous and past requests do not have to be include in the token.

In the immediate future we would like to perform a Technology Acceptance Model study with students to give us feedback on the perceived ease of use of TECH and its perceived usefulness. We would also like to perform a study with a control group of students who have never had any interview prep with those using TECH to see if there is any difference in anxiety during in person.

IX. Limitations

Future iterations of the system can be used for low stake interview practice but the system should not actually replace in-person interviews. The current scoring system may have a Eurocentric bias on how speech is interpreted or generated based on the input text. The prompt may need to be updated when the GPT model is updated.

X. Conclusion

In this paper, we present an overview of the SE interview, how it is currently performed in-person, and how a large language model system like ChatGPT-3 can be engineered to create a tool called TECH for undergraduate computer science students to practice their tech interviewing skills. We also present the feasibility of TECH to provide practice and feedback to the student learner. Through future testing of TECH, we envision the tool will help alleviate the burden from schools that cannot provide their own tech preparation course and give students from all backgrounds the ability to understand how to succeed at a SE interview.

References

[1] "Technology Will Continue to Be the Primary Industry for Wealth Creation Over the Next Decade | Barron's." [Online]. Available: https://www.barrons.com/articles/technology-will-continue-to-be-the-primary-industry-for-wealth-creation-over-the-next-decade-01589454092

[2] "Computer and Information Research Scientists : Occupational Outlook Handbook: : U.S. Bureau of Labor Statistics." [Online]. Available: https://www.bls.gov/ooh/computer-and-information-technology/computer-and-information-research-scientists.htm

[3] M. Behroozi, S. Shirolkar, T. Barik, and C. Parnin, "Debugging Hiring: What Went Right and What Went Wrong in the Technical Interview Process," in *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, Oct. 2020, pp. 71–80.

[4] "Facebook's Hiring Process Hinders Its Effort to Create a Diverse Workforce - Bloomberg." [Online]. Available: https://www.bloomberg.com/news/articles/2017-01-09/facebook-s-hiring-process-hinders-its-effort-to-create-a-diverse-workforce

[5] S. J. Lunn and M. S. Ross, "Cracks in the Foundation: Issues with Diversity and the Hiring Process in Computing Fields," Jul. 2021. [Online]. Available: https://peer.asee.org/cracks-in-the-foundation-issues-with-diversity-and-the-hiring-process-in-computing-fields

[6] P. K. Chua and M. Mazmanian, "Are You One of Us? Current Hiring Practices Suggest the Potential for Class Biases in Large Tech Companies," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW2, pp. 143:1–143:20, Oct. 2020. [Online]. Available: https://doi.org/10.1145/3415214

[7] M. Behroozi, C. Parnin, and T. Barik, "Hiring is Broken: What Do Developers Say About Technical Interviews?" in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct. 2019, pp. 1–9, iSSN: 1943-6106.

[8] S. J. Lunn, M. S. Ross, Z. Hazari, M. A. Weiss, M. Georgiopoulos, K. Christensen, and T. Solis, "Uneven Playing Field: Examining Preparation for Technical Interviews in Computing and the Role of Cultural Experiences," Jul. 2021. [Online]. Available: https://peer.asee.org/uneven-playing-field-examining-preparation-for-technical-interviews-in-computing-and-the-role-of-cultural-experiences

[9] L. Burge, K. Picho-Kiroga, and P. K. Smith, "The Interview Access Gap for Black Engineers," Tech. rep. Seattle, Washington: Karat, 2021. url: https://karat. com/wp . . . , Tech. Rep., 2021.

[10] P. Hall Jr. and K. Gosha, "The Effects of Anxiety and Preparation on Performance in Technical Interviews for HBCU Computer Science Majors," in *Proceedings of the 2018 ACM SIGMIS Conference on Computers and People Research*, ser. SIGMIS-CPR'18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 64–69. [Online]. Available: https://dl.acm.org/doi/10.1145/3209626.3209707

[11] D. Ford, T. Barik, L. Rand-Pickett, and C. Parnin, "The Tech-Talk Balance: What Technical Interviewers Expect from Technical Candidates," in *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 2017, pp. 43–48.

[12] A. Kapoor and C. Gardner-McCune, "Barriers to Securing Industry Internships in Computing," *ACE*, 2020.

[13] A. Alvarez, L. Burge, S. Emanuel, A. Gates, S. Goldman, J. Griffin, H. Keeling, M. J. Madda, B. Okafor, A. Onowho, and G. Washington, "Google tech exchange: an industry-academic partnership that prepares black and latinx undergraduates for high-tech careers," *Journal of Computing Sciences in Colleges*, vol. 35, no. 10, pp. 46–52, Apr. 2020.

[14] L. Burge, M. Mejias, K. Galloway, K. Gosha, and J. Muhammad, "Holistic Development of Underrepresented Students Through Academic: Industry Partnerships," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '17. New York, NY, USA: ACM, 2017, pp. 681–682. [Online]. Available: http://doi.acm.org/10.1145/3017680.3017808

[15] A. N. Washington, L. Burge, M. Mejias, K. Jean-Pierre, and Q. Knox, "Improving Undergraduate Student Performance in Computer Science at Historically Black Colleges and Universities (HBCUs) Through Industry Partnerships," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '15. New York, NY, USA: ACM, 2015, pp. 203–206. [Online]. Available: http://doi.acm.org/10.1145/2676723.2677277

[16] L. Grewe, K. Kanemoto, S. Wang, and J. Espy, "Industry and Academic Collaboration: Google Faculty in Residence Experiences," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, Feb. 2020, pp. 1266–1267. [Online]. Available: https://dl.acm.org/doi/10.1145/3328778.3366984

[17] A. Aziz, T.-H. Lee, and A. Prakash, *Elements of Programming Interviews: The Insiders' Guide*. EPI, 2012, google-Books-ID: y6FLBQAAQBAJ.

[18] G. L. McDowell, *Cracking the Coding Interview: 189 Programming Questions and Solutions*, 6th ed. Palo Alto, CA: CareerCup, Jul. 2015.

[19] J. Mongan, N. S. Kindler, and E. Giguere, *Programming Interviews Exposed: Coding Your Way Through the Interview*, 4th ed. Indianapolis, IN: Wrox, Apr. 2018.

[20] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.

[21] S. Shillsalot, "How you can Jailbreak ChatGPT with these top 4 methods," Apr. 2023. [Online]. Available: https://ambcrypto.com/heres-how-to-jailbreak-chatgpt-with-the-top-4-methods-2/

[22] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–21.