

Will You Trust Me More Than ChatGPT? Evaluating LLM-Generated Code Feedback for Mock Technical Interviews

Swanand Vaishampayan, Chris Brown

Department of Computer Science, Virginia Tech, USA

{swanandsv, dcbrown}@vt.edu

Abstract—Technical interviews are an opportunity for candidates to showcase their technical proficiency to employers. Despite social aspects of tech interviews, preparation for these assessments is often done in isolation. Cooperative preparation methods, such as mock interviews—or simulated interviews between peers—can provide useful feedback and reveal candidates’ technical deficiencies. However, this process is rarely used by candidates in practice. Further, current technical interviews lack authentic feedback for candidates on technical evaluation criteria, such as code correctness, optimality, and complexity. To resolve this, we designed a website to facilitate mock technical interviews and provide users with LLM-generated feedback on code using ChatGPT. We devised a between-subjects study with 46 participants across two settings: human-administered (a live human sharing generated feedback) and automated (direct LLM-generated feedback). Our results show that candidates perceive coding feedback as useful—however, we found participants regard automated feedback as less trustworthy compared to feedback provided by humans. In light of our findings, we discuss implications and guidelines for incorporating trustworthy AI in technical interview feedback systems.

Index Terms—technical interviews, LLM-generated feedback, trustworthy AI, LLM based technical interview feedback, LLM code feedback

I. INTRODUCTION

Technical interviews assess the technical proficiency of candidates pursuing Software Engineering (SE) positions in the tech industry [1]. Technical interviews consist of one or more coding problems, based on data structures and algorithms, to be solved in a fixed time with verbal think-aloud communication in front of an interviewer [2]. However, this process is often criticized for numerous reasons, including their stressful environment [3] and lack of feedback for candidates [4].

The complexity of technical interviews make them difficult to prepare for [5]. Cooperative methods, such as mock interviews where candidates simulate technical interviews with peers, have been shown to significantly enhance preparedness and reduce anxiety [6]. However, these processes are rarely used—with candidates relying on individual preparation with online resources [6], such as LeetCode.¹ In addition, while mock interviews can be valuable, prior work shows peers, such as students, may lack relevant expertise and knowledge [7]. Feedback from peers influences SE in practice [8], and feedback from interviewers providing insights on candidates’ per-

formance could underline deficiencies and enhance preparedness for future assessments [9]. Yet, post-interview feedback is often not provided due to internal policies, legal implications, and lack of organized feedback-giving processes [10]. Further, online resources that provide opportunities for mock interviews with experts can be financially burdensome. For instance, mock interviews with experts on IGotAnOffer² and Interview Buddy³ cost over \$100 per hour.

We speculate large language models (LLMs), machine learning models trained on a large corpus of data, can mitigate these challenge by generating relevant feedback for candidates. LLMs and artificial intelligence (AI)-based systems are increasingly used in hiring processes by employers to streamline candidate evaluations [11], [12] and conduct interview assessments [13], [14]. Prior work also explores the capabilities of LLMs for enhancing interview preparation for candidates across various job domains [15], [16], [17]. LLMs are also increasingly used in SE [18], showing promise for supporting tasks such as code explanations [19], summarization [20], and design [21].

Prior work investigates LLM-generated feedback on code [22], [23]. However, coding in interview settings is unique compared to other contexts, such as classroom and industry settings, due to expectations for demonstrating technical abilities to employers [24] and increased cognitive load [25]. For instance, evaluations for algorithmic-centered technical interviews focus on a variety of factors, including syntactical correctness, logical correctness, style, and optimal time complexity, space complexity, and execution time [26], [27]. In contrast, previous work shows that autograders in computing classes primarily focus on “functional-correctness” [28] and feedback on style and code structure in practice are often missing, requiring manual inspection [28], [29].

To our knowledge, we are the first to study LLMs for generating *technical interview code feedback*. We developed an online system powered by ChatGPT,⁴ a popular LLM released by OpenAI [30], to analyze the quality, usefulness and trustworthiness of LLM-generated feedback on code submitted by users in technical interview preparation. More precisely, we

¹<https://leetcode.com/>

²<https://igotanooffer.com/>

³<https://interviewbuddy.net>

⁴<https://chat.openai.com/>

set out to answer the following research questions (RQs):

RQ1: To what extent do candidates use LLMs to prepare for technical interviews?

RQ2: What is the quality of technical interview coding feedback generated by ChatGPT?

RQ3: How do candidates perceive the usefulness of feedback generated by ChatGPT across human-administered and automated feedback settings?

RQ4: What impact does the source of feedback have on candidates' perceived trust in responses?

We answer the research questions through a controlled between-subjects experiment. Participants ($n = 46$) completed simulated technical interview tasks with our system, receiving feedback across two settings: a) a *human-administered setting*, where a human delivered feedback without participants' knowledge it was LLM-generated; and b) an *automated setting*, where GPT directly provided feedback. Our results show the majority of participants (91%) found LLM-generated feedback useful, demonstrating a willingness to use feedback from LLMs in their technical interview preparation. The quality of feedback, representing the correctness of LLM-generated feedback, was also found to be mostly accurate (83%). However, we found a significant drop in trust for the automated feedback setting compared to the human-administered setting.

Based on our findings, we provide implications for enhancing technical interview preparation with LLM-generated code feedback. In particular, we discuss guidelines for improving Human-AI collaboration in tech interview preparation by promoting quality and trustworthy LLM-generated code feedback to enhance candidates' performance in interviews. Our work makes three contributions to SE literature:

A: We explore the effects and role of LLMs in technical interview preparation. Additionally, we implement a solution to overcome the lack of feedback in technical interviews.

B: We determine the quality and usefulness of LLM-generated feedback on code in technical interview settings. Moreover, we uncover underlying factors influencing candidates' trust in code feedback.

C: We provide guidelines for reliable and trustworthy LLM-generated code feedback in technical interview settings.

II. BACKGROUND AND RELATED WORK

A. Technical Interviews

Technical interviews are a critical part of the tech hiring process. This evaluation method helps hiring managers gauge candidates' technical proficiency, problem-solving skills, communication ability, and cultural fit [31]. Preparation for technical interviews is also difficult for candidates. Prior work [6] found students spend an average of seven hours per week preparing for technical interviews. Hall and Gosha [32] underline anxieties in technical interview preparation faced by CS students from HBCUs. Additional study [3] report increased levels of stress in candidates during technical interviews, resulting in a negative impact on candidates' performance—particularly those identifying as women. Lunn et al. found that preparation for

technical interviews requires special effort from candidates [5], and negative technical interview experiences impact students' computing identity. Our study explores the capabilities of LLMs for providing quality and useful feedback to job seekers on submitted code in tech interview simulations.

B. Trust in AI-Driven Systems

Prior work suggests software engineers distrust LLM-based systems for various SE tasks, such as requirements engineering [33] and code reviews [34]. Further, the 2024 Stack Overflow Developer Survey indicates developers find the lack of trust in output as the largest challenge with working with AI tools [35]. In hiring contexts, Vaishampayan and colleagues [36] show candidates pursuing computing positions distrust AI-based resume parsing systems—widely adopted to streamline hiring processes for companies—preferring humans to review their job application documents. In an effort to improve developer trust in AI systems, Johnson et al. [37] introduce the PICSE framework to identify factors and provide guidelines to developing trustworthy AI-powered development tools. Wang et al. [38] explored contextual factors influencing programmer trust, suggesting enhanced productivity and code quality are key factors for trust in AI code generation tools. Prior work also suggests human-in-the-loop verification can improve ChatGPT-driven requirements engineering [33]. Through this work, we analyze the accuracy and trustworthiness of LLM-generated code feedback for technical interview preparation across two settings: a human-in-the-loop and fully automated approach.

III. SYSTEM DESIGN

A. Mock Interview Platform

We developed a website to simulate mock technical interviews and provide LLM-generated feedback. Mock interviews, or practice technical interviews with peers, are essential for improving preparation for real job assessments [39]. Our website contains three algorithmic coding problems commonly used in technical interviews for participants to select: a) *Longest Common Prefix*; b) *Two Sum*; and c) *Best Time to Buy/Sell Stock*. These problems assess candidates' abilities on a variety of data structures and algorithms, such as strings, arrays, and the sliding window algorithm, which are often covered in CS courses [40] and are popular topics for technical interviews [41]. After selecting a problem, users are presented with the following: (1) the problem statement and constraints, (2) two sample test cases, and (3) a coding environment with functionality to write and compile code. Users are allotted 25 minutes to solve the problem. During pilot studies with seven participants, we verified the problems were of similar difficulty and could be solved within the allotted time.

The website offers two programming languages for users, Java and Python, which are the most common languages candidates select for technical interviews [42]. Users can run and compile the code without any restrictions. The website also incorporates a hint button for users, based on previous work noting candidates often face difficulties starting a coding

problem [32]. The hint provides a textual description of a brute force approach to tackle the problem, without providing any code to users. After developing a solution, the code is submitted manually by users pressing the submit button or automatically submitted after 25 minutes. Then, users are taken to a loading page that provides details about the feedback structure while the ChatGPT API analyzes the code and generates a response for the user. Finally, users are provided with feedback based on the study setting. The website is available online and publicly available for users to access.⁵

B. Mock Interview Feedback

a) *Feedback Generation:* To generate feedback, we use OpenAI's Generative Pre-trained Transformers (GPT) 3.5-turbo model. More modern models, such as GPT-4, have better performance on numerous coding tasks [43]—however we employ the basic model due its its availability and low-cost. While ChatGPT refers to the online chatbot interface built on GPT and our system utilizes the GPT API, for simplicity we refer to the back-end of our system as ChatGPT.

Users of our platform receive automated code feedback generated by ChatGPT on technical interview-specific constructs. Additionally, the feedback contains suggestions to improve upon the constructs, when applicable. For improvement in particular, the feedback contained two sections:

Textual improvement section: Textual improvements consisted of natural language suggestions to enhance code style, code readability, syntax, and the overall approach of the provided code, when applicable.

Code improvement section: Code improvements were only provided if a participant's submitted solution was sub-optimal or contained errors. It consisted of the optimized solution in code form. Figure 1 shows an example, where a participant received suggestions to add input validation.

We used Chain of Thought (CoT) [44] to formulate the system prompt as it improves the code understanding and optimization ability of LLMs [45] to generate better output than zero-shot outputs. The prompt structure is as follows:

Instruction1: "Is the following code correct for the given problem in Java/Python?" <problem statement>
 <user code>
 Instruction2: "Check against test cases " <test cases>
 Instruction3: "Provide detailed feedback regarding time complexity, space complexity, and execution time of the program. Also, suggest improvements for the given code in Java/Python."

Listing 1: Prompt for Feedback Generation

IV. METHODOLOGY

To explore perceptions of LLM-generated feedback on submitted code in technical interview settings, we conducted a study consisting of mock technical interviews with our system.

⁵Blinded for review

A. Study Recruitment

Students at the authors' university were recruited to participate in this study, as they often represent candidates pursuing full-time and intern SE positions. A recruitment email was broadcast through relevant listservs to enlist Computer Science students with technical interviewing experience. A brief survey was shared to schedule interviews, as per participants' availability. We obtained Institutional Review Board (IRB) approval for human subjects research before recruitment.

B. Survey

Before and after the mock technical interview task, participants completed a pre- and post-survey. Surveys are a common method for data collection controlled user studies for software engineering research [46]. The complete survey instruments can be found in the supplemental material.

1) *Pre-interview survey:* The pre-survey contained short answer and open-ended questions targeting the background of participants, such as their industry experience as well as demographic information regarding age, race/ethnicity, and gender. We collected data on their technical interview experience to ensure participants were qualified for the study. Additionally, we asked participants about the resources they use to prepare for technical interviews and if they use LLMs, such as ChatGPT, in preparation processes to answer RQ1.

2) *Post-interview survey:* The post-survey contained short answer, open-ended, and Likert-scale questions to provide insight on the mock interview system and ChatGPT-generated feedback. Participants completed the post-survey after completing the study tasks and observing the feedback in each setting. The post-interview survey questions asked participants about their experience with tool, the usefulness of the feedback, their trust regarding the correctness of the feedback, and their perception of the hint feature.

C. Study Settings

We conducted our study in two different settings: *human-administered* and *automated*. The design choice of selecting two different settings was based on the previous work, which showed end user trust can vary based on the way user perceives inner workings of the system [47]. In both settings participants received ChatGPT-generated feedback. The major difference between the two was the delivery of feedback. In the human-administered setting, a human presented the ChatGPT-generated feedback to participants on their code submission. In the automated setting, participants directly received ChatGPT-generated feedback online and were aware the feedback came from ChatGPT. The submitted code, selected problem and generated feedback were stored for our analysis. We conducted pilot studies ($n = 7$) to assess our study protocol and survey constructs before beginning the study.

Study Task. During the first five minutes of the mock interview, the interviewer gave basic details about the website to participants. After that, participants were given an optional 10 minutes to familiarize themselves with the website. Once the participants were ready, they were emailed the link to

the pre-survey and website. Participants selected one of three problem options. The problems were not disclosed to the participants before the interview. The interviewer supervised the interview for the next 25 minutes while the participants coded solutions for the selected problems. The interviewer moderated the sessions to make sure participants did not use any inappropriate means to solve the problem, such as referring to online resources, copying the solutions from LeetCode, or using generative AI tools. After the code submission, each participant was asked to wait several minutes for the feedback across our two study settings.

1) *Human-administered setting*: In the human-administered feedback setting, we employed a *reverse Wizard of Oz approach* to evaluate participants' trust in feedback mechanisms. In the Wizard of Oz research method, study participants interact with a system user interface which is unknowingly controlled by a human on the back-end [48]. In this study setting, participants interacted with a system and received feedback through human, unaware that the responses were generated by ChatGPT on the back-end. This was done to capture perceived differences of GPT-generated and "human-written" code feedback for tech interview preparation. The first author acted as interviewer and provided the feedback after the each session, and all the interviews were conducted in-person. Figure 2a shows the human-administered feedback setting.

During the waiting period after the code submission, the interviewer acted as if they were reviewing the code to write the feedback. However, the interviewer had access to the back-end of the system, which contained the participants' submitted code and feedback generated by ChatGPT on code complexities and improvements. Human-administered feedback followed a fixed format with headings "Time Complexity, Space Complexity, Execution Time and Improvements" for maintaining consistency in feedback. GPT-generated feedback has obvious cues for participants to detect comments are written by LLMs, such as complex sentence structure, fixed number of words per sentence and ending with "overall", etc., thereby affecting our goal of capturing trust levels of portrayed human-written and LLM feedback. To mitigate this, we induced a "human-touch" to the LLM-generated feedback by applying minor rewording and rephrasing sentences based on the previous research [49] to make AI text non-detectable. While imparting the human-touch, we made sure to maintain the conformity of the original GPT-generated feedback by keeping all the suggested pointers in-tact and by not altering suggestions for improvement in any way. Figure 1 provides an example showing the difference between the original ChatGPT-generated feedback and human-administered version of the same feedback. To emphasize the feedback was human generated, the interviewer emailed the feedback to participants, along with the post-survey link. Participants were instructed to read the feedback first and then fill out the survey.

2) *Automated setting*: The automated feedback setting was identical to the human-administered setting, but with two major differences: A) Participants were explicitly made aware of ChatGPT generated feedback; and B) the feedback was

automatically made available on the same website once the code was submitted. Figure 2b presents the automated feedback setting. After submitting the code, participants had to wait approximately 45 seconds for ChatGPT to generate the feedback before being redirected to a webpage with the feedback outlined and a link to the post-survey.

D. Participants

1) *Human-administered setting*: Since our study aim was to understand participants' perceptions of technical interview code feedback, we used purposive sampling [50], [51] to target students with technical interviewing experience. For the human-administered setting, we interviewed 26 participants. The majority (85%, $n = 23$) had previous technical interview experience. Three participants lacked prior experience and were removed, leaving us with a final sample of ($n = 23$) participants comprising of ($n = 14$) graduate students and ($n = 9$) undergrads. For industry experience, half of the participants 50% ($n = 13$) had internship experience whereas 19% ($n = 5$) had 2-5 years and 12% ($n = 3$) had one year of industry experience or less. 16% ($n = 4$) had no industry experience. 3% ($n = 1$) had 5+ years of industry experience.

2) *Automated setting*: For the automated setting, we conducted mock interviews with 86 participants. Part of the data collection was conducted as a classroom study with 73 students in an undergraduate SE course at the authors' institution. For the in-class interview activity, we made sure participants were exposed to the same conditions as the lab space. The remaining 13 were graduate students who interviewed in the same lab space as the human-administered setting. We excluded candidates who did not have technical interview experience ($n = 36$). This left us with 50 samples. To measure group differences across both study settings, we applied stratified sampling [52] to sample 23 responses to match the distribution of the human-administered setting.

In the final sample, 43% ($n = 10$) were undergraduates and 57% ($n = 13$) were graduates. 9% ($n = 2$) had no industry experience, 48% ($n = 11$) had only internship experience, 17% ($n = 4$) had one or less than one years of experience, 22% ($n = 5$) had 2-5 years of industry experience, and 4% ($n = 1$) had over 5 years of industry experience.

E. Data Analysis

a) *RQ1 & RQ3*: We analyze open-ended responses of participants regarding use of LLMs in preparation of technical interviews and participants' perceptions of the usefulness of code feedback from pre and post-surveys to answer RQ1 and RQ3 respectively. Open-ended questions were analyzed using an open-coding approach based thematic analysis where the two authors independently reviewed and coded the responses. Then, the coders came together to discuss their themes. Disagreements were resolved after the discussion between the coders and consensus were applied in the next iteration of coding. We calculated our inter-rater agreement using Cohen's kappa, where $\kappa = 0.78$ for "substantial" agreement. The data and the codebooks used for analysis are included in our

ChatGPT generated Feedback:

The code provided is correct and returns the maximum profit that can be achieved from buying and selling stocks.

The time complexity of the code is $O(n)$, where n is the length of the prices array. This is because the code iterates through the prices array once to find the maximum profit.

The space complexity of the code is $O(1)$, as it only uses a constant amount of extra space to store the current minimum price and the maximum profit.

The execution time of the program will depend on the input size and the values in the prices array. In the worst case scenario, where `prices.length = 1000000` and all prices are in descending order, the code will take longer to execute.

One improvement that can be made to the code is to handle the case where there is no profit to be made. Currently, if there is no profit to be made, the code returns 0. However, it would be more appropriate to return a negative value to indicate that no profit can be made. This can be done by initializing the `max_profit` variable to `Integer.MIN_VALUE` instead of 0.

Another improvement that can be made is to add input validation to check if the prices array is empty or null. If it is, the code can return an appropriate error message or throw an exception.

Human-administered Feedback:

7. Thanks for participating in the study. Here's your feedback:
The code provided is correct and returns the maximum profit.

Time Complexity:

The time complexity of the code is $O(n)$, where n is the length of the prices array.

Space Complexity:

The space complexity of the code is $O(1)$, as it only uses a constant amount of extra space to store the current minimum price and the maximum profit.

Execution Time:

The execution time of the program will depend on the input size and the values in the prices array. In the worst case scenario, where `prices.length = 1000000` and all prices are in descending order, the code will take longer to execute.

Improvement:

1. Handle the base cases such as no profit.

2. Another improvement that can be made is to add input validation to check if the prices array is empty or null.

Fig. 1: Examples of ChatGPT-generated feedback presented to users in the automated (left) and human-administered (right) study settings.

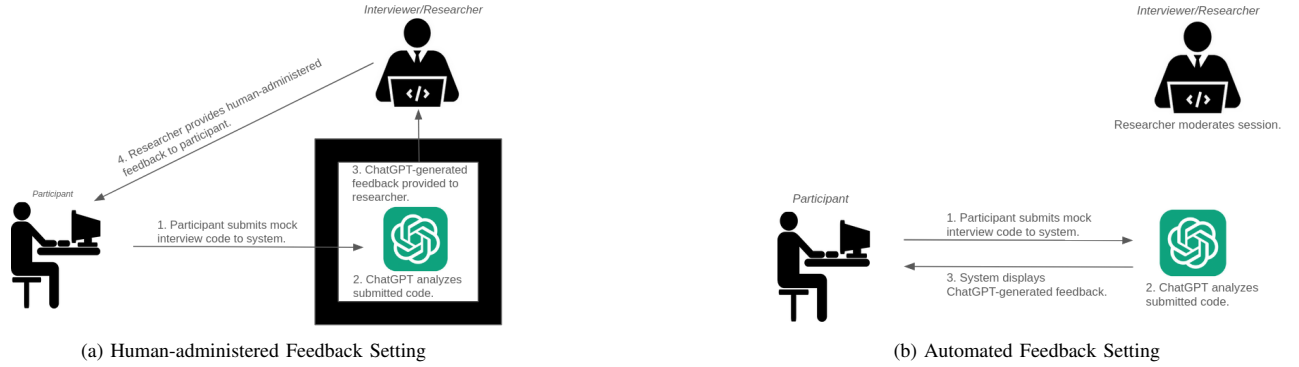


Fig. 2: High-level overview of our two study settings for human-administered and automated LLM-generated feedback.

supplemental materials.

b) RQ2: The quality of ChatGPT generated technical interview feedback is measured by analyzing the feedback generated in both human-administered and automated study settings. In-depth steps of quality analysis are explained in section V-B.

c) RQ4: We analyzed Likert-scale and subsequent open-ended question capturing the trust of participants to answer RQ4. We used a one-tailed t-test to statistically compare the participants' Likert scale responses for trust across each setting. Parametric t-test has prior assumptions such as normality of the data and we verified the normality of both human-administered and automated subgroups using Kolmogorov-Smirnov test of normality [53]. The choice for using the t-test was based on prior work which shows similar power for both parametric and non-parametric tests when analyzing Likert-scale data [54]. In addition, previous studies used this test to compare mean distributions across groups in SE contexts (*i.e.*, [55], [56]) and to measure trust in AI (*i.e.*, [47], [57]).

V. RESULTS

RQ1 and RQ2 depend on both the study settings, therefore we present combined results. For the remaining RQs, we provide separate results for each feedback setting. For quotes, the A-prefix indicates a participant from automated setting while H-denotes a participant from human-administered setting.

A. RQ1: Use of LLMs in preparation

We asked an open-ended question to participants regarding their use of LLMs, such as ChatGPT, in technical interview preparation. We analyzed their responses to understand the reasons for and against their usage.

1) Not seeking GPT help

54% ($n = 25$) of participants reported they do not use LLMs in their preparation. The major themes we discovered are:

No exposure: 56% ($n = 14/25$) of participants noted they have not considered using LLMs for technical interview preparation. For example, A15 responded, “No, using ChatGPT does not reflect the knowledge of the candidate rather an interviewees ability to feed information to something else and can give a false awareness of the subject material”.

Did not exist: 12% ($n = 3$) did not use LLMs as they did not exist when the respondents were preparing for their technical interviews. A16 embodies this in their response, “*I haven’t had a technical interview in the time since chatGPT has come out so I haven’t had a chance to use it to prepare*”.

32% ($n = 8$) of participants reported other reasons, such as the availability of other online resources for technical interview preparation.

2) Seeking GPT help

The remaining 46% ($n = 21$) of participants reported using LLMs for technical interview preparation, providing the following reasons:

Code modifications: 43% ($n = 9$) of the participants noted using LLMs for code modifications, such as asking for optimized code, finding code in different programming languages, etc. For instance, A13 noted, “*I have used it for code optimization*”.

Problem approach: 38% ($n = 8$) referred to LLMs for problem approach-related tasks, such as getting a starting point or querying for a better approach. For instance, A22 mentioned “*I do in order for it to help me breakdown the base concepts in simpler terms or to find patterns I haven’t noticed*”.

Debugging: 19% ($n = 4$) use LLMs for debugging purposes, such as finding syntactical errors in their code. For example, A6 responded “*I interact with chatgpt to find bugs in my code and try to find better ways to solve the problem*”.

B. RQ2: Quality of ChatGPT feedback

We examined the quality of ChatGPT-generated responses in both settings. For the human-administered setting, we analyzed the feedback generated by ChatGPT without human touch. The LLM-generated feedback contained information regarding the correctness, code complexities and suggested improvements (see Section III-B). We evaluated textual and code sections of feedback using the following approach:

Textual section: We manually reviewed each response to determine if it contained correct information regarding the code complexities, correctness and textual improvements when applicable by comparing candidate submitted code and the solution. Since our problems were taken from Leetcode, the optimized solutions were readily available.

Code Improvements: For code improvements, we tested ChatGPT-generated code against the pre-defined test cases provided by LeetCode.

We report feedback is accurate if: a) it passes all textual considerations; and b) the code, when available, passes all test cases.

We analyzed 36 responses (23 from the human-administered setting and 13 from the automated feedback setting) with textual and code improvements. The discrepancy is due to mapping unavailability between participants’ submitted code and survey responses for the class activity in the automated setting. We found the ChatGPT-generated response accuracy was 83.33% ($n = 30$ out of 36). All of the inaccurate responses ($n = 6$) were found to have defects in the textual improvement. We further analyzed the inaccurate feedback

responses to find if there are underlying factors responsible. We found some commonalities:

Inaccuracies in improving time complexities: In two out of six responses, ChatGPT was inaccurate in its improvement about time complexities. Consider the following example feedback generated for a participant (H17):

```
//previous feedback comments
```

```
- Time complexity: The code has a time complexity of O(n) because it iterates through the given array once.
```

```
//more feedback text
```

```
Improvements for the given code: The code can be further optimized by using a single pass through the array instead of two passes. You can use a single loop to check if the difference exists in the seen numbers while adding the current number to the seen numbers.
```

```
//rest of the feedback text ”
```

If we look closely at this example (emphasis added), ChatGPT responded that the time complexity of the submitted code is $O(n)$ as it iterates through the array once—but it also suggests the user can further reduce the complexity by using a single pass which is inaccurate.

Generalized solutions: In four responses, we found ChatGPT feedback did not follow the exact problem statement. Consider a response received by participant A15:

```
//previous feedback comments
```

```
Improvements for the given code: - Another improvement could be to handle the case where there is no solution. The current code always returns [0, 1] as the default output if no solution is found. It would be better to return None or raise an exception in this case to indicate that there is no valid solution.
```

```
//rest of the feedback text ”
```

Despite the problem statement for the Two Sum problem stating “You may assume that each input would have exactly one solution”, ChatGPT generalized the problem and suggested adding an additional check for no solution.

C. RQ3: Usefulness of Feedback

Overall, receiving feedback on technical interview performance yielded a positive response from participants. 91% ($n = 42$) of participants found the feedback to be useful. We further studied the responses to analyze what made the interview feedback useful and found three major themes explaining the participants choice of selection. They are:

Feedback: The LLM-generated code feedback consisted of feedback on code complexities along with suggestions for improvement to underline mistakes and help candidates improve the technical interview code. Participants overall found the feedback useful, such as A1 who noted it “*gave me an optimized and efficient solution to work with*”.

Learning: Candidates’ appreciated the ability to learn from the provided feedback, classified in two types: a) learning about the problem, such as an optimized solution; or b) general coding-related learning, such as tips on improving readability, code documentation etc. For instance, one participant stated “*It*

helped me know that I did in fact have the optimal solution for the problem in terms of time complexity” (H22).

Practice: Candidates felt the website provided them with a similar environment to actual technical interviews, and thus this exercise will help them in preparation of future interviews. For example, one participant stated “It was good practice with the same environment and type of questions” (H12).

We present the findings on feedback usefulness from each setting below.

1) *Human-administered feedback:* 83% of participants ($n = 19$) in the human-administered setting found the overall feedback to be useful. 40% of participants ($n = 9$) reported they received accurate improvement suggestion while 30% ($n = 7$) participants were happy that their mistakes got underlined. 21% of human-administered setting participants ($n = 5$) found the information regarding code complexities *i.e.*, time complexity ($n = 2$), space complexity ($n = 2$) and execution time ($n = 1$) presented in the feedback to be useful. Remaining 9% ($n = 3$) of participants found the feedback layout to be useful. 70% ($n = 16$) reported they learned something new from the feedback. 56% ($n = 9$) learned something new about the problem while remaining 44% ($n = 7$) got tips on general coding improvements. Lastly, 13% ($n = 3$) of candidates were grateful for the simulation practice provided by the website.

2) *Automated feedback:* In the automated setting, 61% of participants ($n = 14$) found the overall feedback to be useful. This was a large drop compared to human-administered setting. The biggest drop was found for accurate improvement suggestion with only 17% ($n = 4$) participants reporting that they received accurate improvement suggestion. 42% ($n = 9$) thought code complexity feedback was helpful while 23% ($n = 5$) and 18% ($n = 3$) reported underlining mistakes and feedback layout to be helpful respectively. There was a decrement observed for new learnings as well when compared with human-administered setting with only 47% ($n = 11$) reporting that they learned something new. For practice, the number remained same when compared with human-administered setting with 13% ($n = 3$) candidates reporting it was good practice.

Overall, we observed a decrease in the perceived usefulness of feedback in the automated setting compared to the human-administered setting. However, upon analyzing with chi-squared test, the difference in usefulness between the settings was found to be non-significant ($X^2 = 0.1723$, $p = 0.9174$).

D. RQ4: Trust in Feedback

a) *Likert Scale:* We asked participants to respond to a 7-point Likert-scale question to understand their trust in feedback. Participants rated the statement “I trust the correctness of the feedback provided” in the post-survey.

1) *Human-administered feedback:* For the human-administered setting, 52% of participants ($n = 12$) mentioned receiving correct feedback and 35% ($n = 8$) got accurate improvements. 17% ($n = 4$) reported receiving help in preparation, while 9% ($n = 2$) trusted the feedback as they

got their mistakes underlined and liked the feedback structure. For distrust, 4% ($n = 1$) of participants reported receiving incorrect textual improvement or defective suggested code.

2) *Automated feedback:* We recorded a decline in participants who reported receiving accurate feedback. For the automated setting, 31% ($n = 7$) participants noted that they received accurate feedback. 31% ($n = 7$) got accurate improvement while 13% ($n = 3$) obtained help in preparation. 9% ($n = 2$) of students got their mistakes underlined and only 4% ($n = 1$) lauded the feedback format. However, despite observing only six instances of inaccurate feedback from ChatGPT overall (see Section V-B), most automated setting participants reported receiving inaccurate feedback. For distrust factors, we saw an increase in partial textual improvement incorrectness (23%, $n = 6$), defective improvement code (4%, $n = 1$), and false prediction (4%, $n = 1$).

Based on responses from each setting, we found the average rating for trust in feedback for the human-administered setting was 6.35—but dropped to 5.48 in the automated setting. We checked the normality of Likert scale distributions using the Kolmogorov-Smirnov [53] test before applying parametric one-tailed t-test. Both human-administered ($D = 0.2657$, $p = 0.0635$) and automated ($D = 0.1839$, $p = 0.3718$) settings were found to be normal, meeting the normality assumption of t-tests. **Our results found a statistically significant difference ($t = 2.854$, $p = 0.003$), indicating a major decrease in trust for feedback delivered through an automated system compared to a human response.**

b) *Open-ended:* Following the Likert scale questions, we asked participants to justify their rating. From the open-ended responses, we derived five themes regarding participants’ trust in technical interview feedback from both settings:

Correctness: Problem logic was checked for correctness to judge whether submitted code could solve the problem and the provided test cases along with code complexities. We found ($n = 12$) such instances reported in human-administered setting and ($n = 7$) in automated setting.

Accurate suggestions: In improvement suggestions, feedback contained recommendations regarding overall style, syntax errors, optimal coding solutions, etc. The LLM-generated feedback should state all the improvement suggestions accurately along with improved code if applicable. Accurate suggestions were reported in ($n = 8$) responses in human-administered and ($n = 7$) responses in automated settings.

Finding mistakes: Candidates underlined the feedback presented mistakes in submitted code and then provided corrections for them. We found this theme in ($n = 2$) responses each from both settings.

Helpful for preparation: ($n = 4$) from human-administered and ($n = 3$) candidates from automated setting believed the website could be helpful in preparation of technical interviews

Feedback format: ($n = 2$) from human-administered setting and ($n = 1$) candidate from automated setting appreciated the overall structure of feedback, the depth of details provided in the feedback.

On the other hand, we found three factors driving participants' distrust of interview feedback:

False prediction: Participants submitted code with errors, such as logic or syntax errors. However, the LLM-generated feedback did not catch them, reporting the code was correct. This theme was only reported by one participant in the automated setting.

Partial incorrect textual improvement: The textual suggestions were not entirely accurate. For example making false improvement suggestions etc. This theme was reported in ($n = 2$) responses in human-administered setting and ($n = 6$) for the automated setting.

Defective improvement code: One participant each from the human-administered and automated settings mentioned that the suggested improved code contained bugs or failed to implement the correct logic.

Example quotes are presented in Table I. The extended codebook can be found in our supplemental data.

VI. DISCUSSION

Our RQ3 results (section V-C) suggest technical interview coding feedback is useful for candidates. Participants showed overwhelming support ($n = 91\%$) for our system, strengthening the motivation for future technical interview feedback systems. RQ2 results (section V-B) indicate the LLM-generated responses were accurate. However, based on RQ4 analysis (section V-D) trust levels significantly dropped for automated setting when compared with human-administered. While technical interview code feedback is valuable, the only feedback provided in current technical interview settings is a job offer or not—and thus lacks sufficient information for candidates' to improve their skills. Further, interviewers are wary of providing feedback [10] and peers in mock interviews have different expectations and lack the expertise of actual interviewers [24]. Thus, we provide implications for designing accurate and trustworthy tools for human-AI collaboration to enhance code feedback, in particular for technical interviews.

A. Feedback Quality

RQ2 results (section V-B) showed ChatGPT-generated feedback was mostly accurate (83.33%). When provided, the ChatGPT-generated feedback presented as code was 100% accurate. This aligns with the previous work [58], where ChatGPT is found to be effective in solving LeetCode Easy and Medium level questions. The problems used in our study fall in this same category of problems.

However, all of the inaccurate responses ($n = 6$) were found to have defects in the textual improvement feedback provided to users as natural language. Among the inaccurate responses for textual improvement suggestions, two were for inaccuracies where feedback suggested inaccurate time complexity improvements. The remaining inaccuracies ($n = 4$) were for generalized solutions, where ChatGPT-generated feedback did not follow the given problem constraints. We analyzed both of these inaccuracies for finding the root cause of these mistakes.

Inaccurate Time Complexity improvements: Code optimization [59] aims to determine where performance improvements could be achieved by reducing the time complexity of code. Previous work found ChatGPT to be effective for static code analysis [44], such as predicting the time complexity of source code [60]. However, static analysis via ChatGPT is not always perfect [44] and is prone to errors. For example, in Section V-B we report an example of participant H17 receiving wrong feedback for improvement in time complexity.

Guideline: Human-AI collaboration Candidates cannot solely rely on fellow peers to provide mock interview feedback, as they have different expectations [24] and can lack expertise in the area. They also cannot rely only on AI, which can provide inaccurate code feedback. However, candidate cooperation with AI may be able to detect and correct inaccuracies in GPT generated feedback. Prior work shows positive results leveraging Human-AI collaboration to augment SE-related tasks, such as code migration [61] and generation [62]. Thus, incorporating functionality, such as conversational programming and problem-solving [63], can support candidate cooperation with AI to enhance the quality of LLM-generated feedback in technical interview settings rather than using AI systems as a black box [64].

Generalized solutions: Upon closer inspection, we found GPT hallucinations as the main reason for ChatGPT overgeneralizing solutions. All LLMs suffer from "hallucinations", where generative models output inaccurate and inconsistent responses [65], [66]. For example, participant A10 reported of receiving a feedback to write test cases to improve the code writing when question did not ask for it specifically.

"feedback mentioned to write test cases, but it was not part of the problem statement. It was not fully correct." (A10)

For our study, prompt engineering techniques, particularly CoT, were used to formulate prompts for LLM-generated tech interview code feedback. However, more advanced techniques can enhance prompts to mitigate generalization issues.

Guideline: Provide Prompt Engineering Context The remedy for minimizing hallucinations is providing more "higher quality context" to LLMs through prompt engineering. For example, previous work on Retrieval-Augmented Generation (RAG) [67] recommends prompt engineering methods, such as tapping into external knowledge bases apart from training data, to minimize hallucinations for downstream tasks such as Q&A, Natural Machine Translation, and Summarization [68]. Further, re-injecting problem constraints in prompts can reinforce context to enhance the accuracy and relevance of LLM-generated code feedback in technical interview settings.

B. Trust and Usefulness

We define trust based on the work by Lee et al. [69], as the "attitude that an agent will help achieve an individual's goals in a situation characterized by uncertainty and vulnerability". Our RQ4 findings indicate that trust-influencing factors for technical interview feedback are code correctness, finding mistakes, feedback format and accurate improvements. Each of these factors are based on the output of the system. This

TABLE I: Feedback Trust Comments

Construct	Human-administered Feedback	Automated Feedback
<i>Trust</i>		
Correctness	“The feedback seemed correct and it also gave me useful improvements” (H14)	“I think the tool was very accurate” (A19)
Finding Mistakes	“The feedback was helpful. It gave detailed information regarding my mistake and also gave me an updated code with the most optimal approach which was O(N) using a hashmap” (H3)	“The feedback showed me my rookie mistake. Need to brush up my coding” (A6)
Feedback Format	“The interviewer gave the correct feedback. It provided me with some useful tips for the future improvement. I liked the feedback format as well.” (H18)	“It had good feedback and feedback was descriptive” (A22)
Accurate Improvement	“The feedback seemed correct and it also gave me useful improvements” (H8)	“satisfying feedback. I liked the suggestion about the trie structure. That could reduce the number of comparisons” (A3)
Helpful in Preparation	“I gave 6 because the feedback was correct and helpful. If this practice is followed in an actual interview it will help to prepare better.” (H2)	“I learned a lot. Feedback had working code and a detailed description of approach. Glad I participated here!” (A7)
<i>Distrust</i>		
False Prediction	N/A	“I think the program was wrong. But the feedback said it was correct. Even though the improvement section was good, it was not accurate” (A1)
Defective Improvement Code	“One of the feedback bullet points didn’t make sense for me. 3. The line ‘return [seen[diff], i]’ should be ‘return [seen[diff], i]’ to return the indices of the two numbers that add up to the target. It looks like the exact same line. Other than that, everything else was helpful.” (H9)	“There were some problems with the ‘correct’ code and it wasn’t entirely accurate.” (A17)
Partial Incorrect Textual Improvement	“I agree with the most part mentioned in feedback that’s why I gave a 6. However I do not agree with the improvement regarding the separate check if array only has 2 values.” (H1)	“I deducted a point as I do not understand the 3rd improvement point. My current solution is already O(n), the improvement that was suggested also gives the solution in O(n). More details on this point would’ve been great.” (A11)

differs from past literature (see Section II-B), which found that trust was influenced by interpretability factors, or how and why a certain decision is made by machine learning models, such as understanding, explainability and transparency [70], [71], [72]. Additionally, these also differ from user behavior-trust relation found in previous works [73], [74] and productivity-trust relation found in coding context by Wang et al. [38]. We posit this is due to differences between technical interview feedback constructs and other code related feedback. Interview feedback contains textual and code sections. Feedback generation is a complex process which requires static analysis to find syntactical errors, time complexity, etc. along with code optimization and further ways to improve code. Previous AI-assisted code systems mainly focus on one construct at a time—for instance, code generation with GitHub Copilot [38].

We observed a significant reduction in trust for the automated feedback setting compared to the human-administered feedback setting (see Section V-D). This is in accordance with previous work showing a lack of trust in AI-driven systems used for hiring [36]. We posit this is due to the ‘impact’ of the feedback if similar system is used in actual hiring. For example, in healthcare domains humans trust human decisions more than algorithmic decisions, regardless the accuracy level of the AI decisions [75]. Similarly, while ChatGPT-generated output was mostly accurate, participants in our study reported less trust for feedback in the automated setting.

Guideline: User ‘trust’ is dynamic Care should be taken in designing the feedback systems, as trust varies according to

domain of use, output, end-user expectations and context of use. For instance, candidates from minority backgrounds face unique challenges in technical interview preparation [32]. We further analyzed factors of distrust to explore the remedies of these issues in an effort to improve user trust in LLM-generated code feedback for technical interviews.

Mistakes in ChatGPT-generated feedback: The inaccurate feedback was one of the main factors behind the decline of trust. The same mistakes were also found in the human-administered feedback—but since the feedback came from a human, candidates trusted the “human-face value”. The following participant response embodies this sentiment:

“The feedback was up to the point. Interviewer told me he has some test cases in background and my code will run against those so I trust his words. He also gave me pretty compact and helpful feedback” (H24)

This finding contradicts previous work [76] which suggests AI assistance causes over-reliance on AI generated output, causing end users to overlook the AI made mistakes. Our study witnessed the opposite, where AI-generated responses in automated setting were minutely checked for every detail, with more perceived inaccuracies than actual errors, while human-administered feedback went unchecked. In our study, participants did not blindly accept the AI suggested feedback.

RQ3 results indicate a similar trend for usefulness with automated setting participants finding feedback less useful than human-administered setting. The biggest factor responsible for this was a rise in perceived inaccurate feedback.

Since automated feedback went through more minute checking by participants, with more reported inaccuracies, automated LLM-generated was deemed less useful from candidates' perspectives. Therefore, future systems should demonstrate the accuracy of feedback to avoid perceived errors, improve its usefulness, and ultimately enhance trust.

Guideline: Incorporate mechanisms to check feedback correctness The use of LLMs in generating mock interview feedback can be effective for enhancing candidates' preparation, but that feedback may have inaccuracies. Therefore, care should be taken when using LLMs to generate interview feedback. To ensure quality feedback, future systems can take the following measures:

a) *Audit checks:* Regular audit checks can ensure the quality and correctness of feedback. Previous work [77] recommends expert audits of ML-based hiring systems to ensure quality and fairness in outcomes. Similar approaches can leverage experienced developers to assess the output of LLM-generated code feedback for technical interview preparation.

b) *End-user feedback:* Another option is providing opportunities for end-users to report feedback [78], [79]. For instance, candidates using LLMs for interview practice can report cases of inaccurate results. Feedback from end users can underline deficiencies in LLM-generated code feedback, which could be resolved to enhance tech interview preparation.

General Distrust in Machine Learning: In the automated feedback setting, candidates were aware that the feedback was generated by ChatGPT. Research shows general human distrust towards deep learning-based systems as they are a "black box" in nature [80]. We also encountered this in our study, with an example response given below:

"I do not trust ml models in general, they make a lot of mistakes and are biased sometimes." (A21)

Guideline: Human-in-the-loop designs User trust in AI drives the acceptability of AI systems [81]. Our study results indicate that candidates trust human-administered feedback substantially more than fully automated feedback. To enhance trust in AI-based mock interview feedback systems, we posit incorporating Human-AI collaboration approaches, such as human-in-the-loop functionality. For instance, in mock technical interview settings systems could consider incorporating functionality where two candidates, one "interviewer" and one "interviewee", can practice together— and the generated feedback can be filtered and administered by the mock interviewer. This collaboration could be synchronous, with live collaboration between interviewer and interviewee in-person or virtual, or asynchronous without a partner present to improve skills needed for assessments [82].

VII. LIMITATIONS AND FUTURE WORK

While our study sample for both experiment settings was diverse, we only interviewed graduate and undergraduate students from one institution. This sample does not represent all candidates seeking SE positions, such as experienced software professionals in job transition. We use human perception

to evaluate the quality of ChatGPT feedback, which may incorporate bias. However, this reflects real interview settings where human judgment is used to evaluate candidates' performances. The majority of technical interviews constitute a data structures-oriented coding question, but it might not be true for every technical interview. We mitigated this by selecting relevant challenges actively used in interviews. A threat to the construct validity of this work is the modified ChatGPT-generated feedback in the human-administered setting. This prevented participants from knowing the feedback was generated by ChatGPT [49]—however, this could impact their overall perceptions of the feedback. To mitigate this, one researcher conducted all of the human-administered studies and did not make major alterations to the original feedback.

Finally, a threat to the internal validity of our findings is participants' inability to complete the coding task in 25 minutes—triggering frustration and leading to reduced trust levels on the code feedback. We mitigated this through the pilot study where we examined the difficulty of coding problems and verified they were of same difficulty and could be solved within allotted time. Participants in both study settings were provided with the same sample of problems. Additionally, during the initial instruction period for both settings the interviewer encouraged participants to treat the study as an actual tech interview, demonstrating problem-solving abilities and technical approaches beyond the accuracy of given solutions [26].

Future work can explore LLM-generated feedback for other aspects, such as incorporating speech-to-text capabilities to input think-aloud transcripts into ChatGPT for additional feedback on interview performance. Moreover, we aim to explore socio-technical solutions to facilitate collaborative computational thinking and verbal communication for enhancing tech interview preparation and performance with LLMs. Such systems can further be extended for actual use in real tech interview scenarios to explore the capabilities of human-centric AI-based tools for improving the tech hiring pipeline.

VIII. CONCLUSION

Technical interviews are an inseparable part of tech hiring processes. Current technical interview processes have numerous flaws, such as a lack of feedback on candidates' interview performance. This lack of feedback can increase stress and frustration for job seekers in technical interview preparation, degrading their performance in actual hiring assessments. To that end, we developed a platform to provide LLM-generated feedback on interview practice code. We evaluated the usefulness and quality of responses and candidates' trust of interview feedback in human-administered and automated feedback settings. We show participants found LLM-generated feedback useful and accurate, but candidates' trust in the automated setting significantly declined compared to feedback provided by a human. Based on our findings, we provide implications for enhancing the quality and trustworthiness of LLM-generated feedback systems and guidelines to improve AI-driven systems in the context of technical interviews.

REFERENCES

- [1] T. Kaatz, "Hiring in the software industry," *IEEE Software*, vol. 31, no. 6, pp. 96–96, 2014.
- [2] A. Kapoor and C. Gardner-McCune, "Introducing a technical interview preparation activity in a data structures and algorithms course," in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 2*, pp. 633–634, 2021.
- [3] M. Behroozi, S. Shirolkar, T. Barik, and C. Parnin, "Does stress impact technical interview performance?," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 481–492, 2020.
- [4] M. Behroozi, S. Shirolkar, T. Barik, and C. Parnin, "Debugging hiring: What went right and what went wrong in the technical interview process," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*, pp. 71–80, 2020.
- [5] S. J. Lunn, E. Zerbe, and M. S. Ross, "Need for change: How interview preparation and the hiring process in computing can be made more equitable," in *2022 CoNECD (Collaborative Network for Engineering & Computing Diversity)*, 2022.
- [6] B. A. Bell, "Understanding the preparation phase of technical interviews," Master's thesis, Virginia Tech, 2023.
- [7] J. Carver, L. Jaccheri, S. Morasca, and F. Shull, "Issues in using students in empirical studies in software engineering education," in *Proceedings. 5th international workshop on enterprise networking and computing in healthcare industry (IEEE Cat. No. 03EX717)*, pp. 239–249, IEEE, 2004.
- [8] R. Sach and M. Petre, "Feedback: How does it impact software engineers?," in *2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering (CHASE)*, pp. 129–131, 2012.
- [9] E. L. Vargas, J. Hejderup, M. Kechagia, M. Bruntink, and G. Gousios, "Enabling real-time feedback in software engineering," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, pp. 21–24, 2018.
- [10] S. Iyer, "Why tech companies don't give feedback to job interview candidates," *Interview Kickstart*, 2024. <https://www.interviewkickstart.com/blogs/articles/top-reasons-why-tech-companies-dont-give-feedback-to-job-interview-candidates>.
- [11] J. Hu, "Over 99% of fortune 500 companies use applicant tracking systems (ats)," *Jobscan Blog*, 2019. <https://www.jobscan.co/blog/99-percent-fortune-500-ats/>.
- [12] L. Li, T. Lassiter, J. Oh, and M. K. Lee, "Algorithmic hiring in practice: Recruiter and hr professional's perspectives on ai use in hiring," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 166–176, 2021.
- [13] L. Hickman, R. Saef, V. Ng, S. E. Woo, L. Tay, and N. Bosch, "Developing and evaluating language-based machine learning algorithms for inferring applicant personality in video interviews," *Human Resource Management Journal*, vol. 34, no. 2, pp. 255–274, 2024.
- [14] J. H. Thakkar, C. Thomas, and D. B. Jayagopi, "Automatic assessment of communication skill in real-world job interviews: A comparative study using deep learning and domain adaptation," in *Proceedings of the Fourteenth Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 1–11, 2023.
- [15] Y.-C. Chou, F. R. Wongso, C.-Y. Chao, and H.-Y. Yu, "An AI mock-interview platform for interview performance analysis," in *2022 10th International Conference on Information and Education Technology (ICIET)*, IEEE, Apr. 2022.
- [16] M. Li, X. Chen, W. Liao, Y. Song, T. Zhang, D. Zhao, and R. Yan, "Ezinterviewer: To improve job interview performance with mock interview generator," 2023.
- [17] S. Y. Liaw, J. Z. Tan, S. Lim, W. Zhou, J. Yap, R. Ratan, S. L. Ooi, S. J. Wong, B. Seah, and W. L. Chua, "Artificial intelligence in virtual reality simulation for interprofessional communication training: mixed method study," *Nurse education today*, vol. 122, p. 105718, 2023.
- [18] W. Mendes, S. Souza, and C. De Souza, "you're on a bicycle with a little motor": Benefits and challenges of using ai code assistants," in *Proceedings of the 2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering*, pp. 144–152, 2024.
- [19] E. Chen, R. Huang, H.-S. Chen, Y.-H. Tseng, and L.-Y. Li, "Gptutor: a chatgpt-powered programming tool for code explanation," *arXiv preprint arXiv:2305.01863*, 2023.
- [20] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, "Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design," *arXiv preprint arXiv:2303.07839*, 2023.
- [21] A. Ahmad, M. Waseem, P. Liang, M. Fahmideh, M. S. Aktar, and T. Mikkonen, "Towards human-bot collaborative software architecting with chatgpt," in *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pp. 279–285, 2023.
- [22] H. Gabbay and A. Cohen, "Combining llm-generated and test-based feedback in a mooc for programming," in *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, pp. 177–187, 2024.
- [23] M. Watanabe, Y. Kashiwa, B. Lin, T. Hirao, K. Yamaguchi, and H. Iida, "On the use of chatgpt for code review: Do developers like reviews by chatgpt?," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pp. 375–380, 2024.
- [24] 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)*, pp. 43–48, IEEE, 2017.
- [25] M. Behroozi, A. Lui, I. Moore, D. Ford, and C. Parnin, "Dazed: measuring the cognitive load of solving technical interview problems at the whiteboard," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, pp. 93–96, 2018.
- [26] G. L. McDowell, *Cracking the coding interview: 189 programming questions and solutions*. CareerCup, LLC, 2015.
- [27] "tech-interview-handbook." <https://www.techinterviewhandbook.org/coding-interview-rubrics/>.
- [28] L. Saliba, E. Shioji, E. Oliveira, S. Cohnery, and J. Qi, "Learning with style: Improving student code-style through better automated feedback," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pp. 1175–1181, 2024.
- [29] E. Van Den Aker and E. Rahimi, "Design principles for generating and presenting automated formative feedback on code quality using software metrics," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, pp. 139–150, 2024.
- [30] S. Mollman, "Chatgpt gained 1 million users in under a week. here's why the ai chatbot is primed to disrupt search as we know it," *yahoo! finance*, vol. 9, 2022.
- [31] 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)*, pp. 1–9, IEEE, 2019.
- [32] 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*, pp. 64–69, 2018.
- [33] N. Marques, R. R. Silva, and J. Bernardino, "Using chatgpt in software requirements engineering: A comprehensive review," *Future Internet*, vol. 16, no. 6, p. 180, 2024.
- [34] M. Vijayvergiya, M. Salawa, I. Budiselić, D. Zheng, P. Lamblin, M. Ivanković, J. Carin, M. Lewko, J. Andonov, G. Petrović, et al., "Ai-assisted assessment of coding practices in modern code review," in *Proceedings of the 1st ACM International Conference on AI-Powered Software*, pp. 85–93, 2024.
- [35] "2024 developer survey," *Stack Overflow*, 2024. <https://survey.stackoverflow.co/2024/>.
- [36] S. Vaishampayan, S. Farzanehpour, and C. Brown, "Procedural justice and fairness in automated resume parsers for tech hiring: Insights from candidate perspectives," in *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 103–108, IEEE, 2023.
- [37] B. Johnson, C. Bird, D. Ford, N. Forsgren, and T. Zimmermann, "Make your tools sparkle with trust: The picse framework for trust in software tools," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 409–419, IEEE, 2023.
- [38] R. Wang, R. Cheng, D. Ford, and T. Zimmermann, "Investigating and designing for trust in ai-powered code generation tools," *arXiv preprint arXiv:2305.11248*, 2023.
- [39] B. Spence and A. Russell, "Using mock interviews to improve students' resume and interview skills," *Radiologic technology*, vol. 94, no. 4, pp. 296–298, 2023.

- [40] M. Lam, "Data structures and algorithms," 2015.
- [41] J. Paul, "interview topics," 2021. <https://medium.com/javarevisited/10-essential-topics-and-resources-for-coding-and-program-interviews-4017cac3a522/>.
- [42] Y. Tay, "Which programming language to use for coding interviews," *Tech Interview Handbook*, 2023. <https://www.techinterviewhandbook.org/programming-languages-for-coding-interviews/>.
- [43] M. Lewandowski, P. Łukowicz, D. Świątlik, and W. Barańska-Rybak, "An original study of chatgpt-3.5 and chatgpt-4 dermatological knowledge level based on the dermatology specialty certificate examinations," *Clinical and Experimental Dermatology*, p. llad255, 2023.
- [44] C. Liu, X. Bao, H. Zhang, N. Zhang, H. Hu, X. Zhang, and M. Yan, "Improving chatgpt prompt for code generation," *arXiv preprint arXiv:2305.08360*, 2023.
- [45] Q. Xu, D. Yang, and L. Zhang, "Code optimization chain-of-thought: Structured understanding and self-checking," in *Proceedings of the 2024 4th International Conference on Artificial Intelligence, Big Data and Algorithms*, pp. 425–430, 2024.
- [46] A. J. Ko, T. D. LaToza, and M. M. Burnett, "A practical guide to controlled experiments of software engineering tools with human participants," *Empirical Software Engineering*, vol. 20, pp. 110–141, 2015.
- [47] R. Kocielnik, S. Amershi, and P. N. Bennett, "Will you accept an imperfect ai? exploring designs for adjusting end-user expectations of ai systems," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2019.
- [48] N. Dahlbäck, A. Jönsson, and L. Ahrenberg, "Wizard of oz studies: why and how," in *Proceedings of the 1st international conference on Intelligent user interfaces*, pp. 193–200, 1993.
- [49] S. Mitrović, D. Andreoletti, and O. Ayoub, "Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text," *arXiv preprint arXiv:2301.13852*, 2023.
- [50] I. Etikan, S. A. Musa, R. S. Alkassim, et al., "Comparison of convenience sampling and purposive sampling," *American journal of theoretical and applied statistics*, vol. 5, no. 1, pp. 1–4, 2016.
- [51] M. Q. Patton, "Qualitative research and evaluation methods. thousand oaks," *Cal.: Sage Publications*, vol. 4, 2002.
- [52] V. L. Parsons, "Stratified sampling," *Wiley StatsRef: Statistics Reference Online*, pp. 1–11, 2014.
- [53] Z.-x. Liu and H. Chen, "Power analysis for testing two independent groups of likert-type data," in *2015 5th International Conference on Computer Sciences and Automation Engineering (ICCSAE 2015)*, pp. 34–39, Atlantis Press, 2016.
- [54] J. C. De Winter and D. Dodou, "Five-point likert items: t test versus mann-whitney-wilcoxon," *Practical assessment, research & evaluation*, vol. 15, no. 11, p. 11, 2010.
- [55] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd international conference on software engineering*, pp. 1–10, 2011.
- [56] H. Mumtaz, C. Paradis, F. Palomba, D. A. Tamburri, R. Kazman, and K. Blincoe, "A preliminary study on the assignment of github issues to issue commenters and the relationship with social smells," in *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering*, pp. 61–65, 2022.
- [57] C. Qian and J. Wexler, "Take it, leave it, or fix it: Measuring productivity and trust in human-ai collaboration," in *Proceedings of the 29th International Conference on Intelligent User Interfaces*, pp. 370–384, 2024.
- [58] N. Nathalia, A. Paulo, and C. Donald, "Artificial intelligence vs. software engineers: An empirical study on performance and efficiency using chatgpt," in *Proceedings of the 33rd Annual International Conference on Computer Science and Software Engineering*, pp. 24–33, 2023.
- [59] E. Bodden, "Self-adaptive static analysis," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, pp. 45–48, 2018.
- [60] S. MacNeil, A. Tran, D. Mogil, S. Bernstein, E. Ross, and Z. Huang, "Generating diverse code explanations using the gpt-3 large language model," in *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2*, pp. 37–39, 2022.
- [61] B. Omidvar Tehrani and A. Anubhai, "Evaluating human-ai partnership for llm-based code migration," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pp. 1–8, 2024.
- [62] S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA1, pp. 85–111, 2023.
- [63] J. Z.-P. L. Qi, B. Hartmann, and J. D. N. Norouzi, "Conversational programming with llm-powered interactive support in an introductory computer science course," 2023.
- [64] D. Wang, E. Churchill, P. Maes, X. Fan, B. Shneiderman, Y. Shi, and Q. Wang, "From human-human collaboration to human-ai collaboration: Designing ai systems that can work together with people," in *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pp. 1–6, 2020.
- [65] Y. Chen, Q. Fu, Y. Yuan, Z. Wen, G. Fan, D. Liu, D. Zhang, Z. Li, and Y. Xiao, "Hallucination detection: Robustly discerning reliable answers in large language models," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM '23*, (New York, NY, USA), p. 245–255, Association for Computing Machinery, 2023.
- [66] V. Rawte, S. Chakraborty, A. Pathak, A. Sarkar, S. Tonmoy, A. Chadha, A. P. Sheth, and A. Das, "The troubling emergence of hallucination in large language models—an extensive definition, quantification, and prescriptive remediations," *arXiv preprint arXiv:2310.04988*, 2023.
- [67] H. Kang, J. Ni, and H. Yao, "Ever: Mitigating hallucination in large language models through real-time verification and rectification," *arXiv preprint arXiv:2311.09114*, 2023.
- [68] S. Tonmoy, S. Zaman, V. Jain, A. Rani, V. Rawte, A. Chadha, and A. Das, "A comprehensive survey of hallucination mitigation techniques in large language models," *arXiv preprint arXiv:2401.01313*, 2024.
- [69] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Human factors*, vol. 46, no. 1, pp. 50–80, 2004.
- [70] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.
- [71] A. Erasmus, T. D. Brunet, and E. Fisher, "What is interpretability?," *Philosophy & Technology*, vol. 34, no. 4, pp. 833–862, 2021.
- [72] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.
- [73] T. Bridgwater, M. Giuliani, A. van Maris, G. Baker, A. Winfield, and T. Pipe, "Examining profiles for robotic risk assessment: Does a robot's approach to risk affect user trust?," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 23–31, 2020.
- [74] B. F. Yuksel, P. Collisson, and M. Czerwinski, "Brains or beauty: How to engender trust in user-agent interactions," *ACM Transactions on Internet Technology (TOIT)*, vol. 17, no. 1, pp. 1–20, 2017.
- [75] M. K. Lee and K. Rich, "Who is included in human perceptions of ai?: Trust and perceived fairness around healthcare ai and cultural mistrust," in *Proceedings of the 2021 CHI conference on human factors in computing systems*, pp. 1–14, 2021.
- [76] Z. Bućinca, M. B. Malaya, and K. Z. Gajos, "To trust or to think: cognitive forcing functions can reduce overreliance on ai in ai-assisted decision-making," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–21, 2021.
- [77] L. Yarger, F. Cobb Payton, and B. Neupane, "Algorithmic equity in the hiring of underrepresented it job candidates," *Online information review*, vol. 44, no. 2, pp. 383–395, 2020.
- [78] J. O. Johanssen, A. Kleebaum, B. Bruegge, and B. Paech, "How do practitioners capture and utilize user feedback during continuous software engineering?," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 153–164, IEEE, 2019.
- [79] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE international requirements engineering conference (RE)*, pp. 125–134, IEEE, 2013.
- [80] W. J. von Eschenbach, "Transparency and the black box problem: Why we do not trust ai," *Philosophy & Technology*, vol. 34, no. 4, pp. 1607–1622, 2021.
- [81] S. Kelly, S.-A. Kaye, and O. Oviedo-Trespalacios, "What factors contribute to acceptance of artificial intelligence? a systematic review," *Telematics and Informatics*, p. 101925, 2022.
- [82] M. Behroozi, C. Parnin, and C. Brown, "Asynchronous technical interviews: reducing the effect of supervised think-aloud on communication ability," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 294–305, 2022.