"I Value LeetCode Over My Coursework": CS Students' Preparation Strategies and Perceptions of Technical Interviews

Daniel Manesh*
danielmanesh@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

Teresa Thomas*
teresa2020@vt.edu
Virginia Tech
Blacksburg, Virginia, Location

Chris Brown
dcbrown@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

Sang Won Lee
sangwonlee@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

Abstract

Technical job interviews for software engineering positions frequently require live problem-solving under observation. These high-stakes interviews are challenging for students, as they require them to prepare for the interview separately. This study investigates how computer science (CS) students prepare for technical interviews and how they perceive these interviews in relation to the CS curriculum and industry job expectations. To that end, we interviewed n=20 students and recent graduates with recent technical interview experience. We found that most relied heavily on repeated practice with coding problems, often drawn from widely circulated problem sets. They also leveraged their social capital by working with peers during interview preparation. Many participants reported a disconnect between interview content and their formal coursework, leading some to prioritize interview preparation over academic responsibilities. Participants expressed a desire for interview formats that better reflect real-world job tasks. These findings offer insights into how academic programs and employers might better support the equitable and effective assessment of job candidates.

CCS Concepts

• **Social and professional topics → Employment issues**.

Keywords

Technical Interviews, Coding Interviews, Job Search

ACM Reference Format:
Daniel Manesh, Teresa Thomas, Chris Brown, and Sang Won Lee. 2026. "I Value LeetCode Over My Coursework": CS Students' Preparation Strategies and Perceptions of Technical Interviews. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-SEET '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3786580.3786991

*Both authors contributed equally to this research.

1 Introduction

Hiring for software engineering roles commonly involves a process known as the *technical interview*. In these interviews, candidates are typically asked to solve programming problems by writing code on the spot, either on a whiteboard or in an online editor (e.g., Google Docs, CollabEdit). Throughout the exercise, candidates are expected to think aloud about their approach and reasoning, enabling interviewers to assess both their problem-solving processes and communication skills [48, 56]. This method remains a standard practice in the industry for evaluating whether applicants possess the technical expertise and interpersonal abilities essential for contributing to collaborative, large-scale software development [34].

However, the setting in which a job candidate is required to program on the spot during a technical interview differs significantly from typical programming environments [4, 7]. Candidates are generally advised to invest a significant amount of time preparing for technical interviews [17], oftentimes with no institutional support. In this context, computer science (CS) students may feel confused: while technical interview topics, such as data structures and algorithms, are covered in CS courses, these classes rarely aim to prepare them for effective interview performance. Moreover, what they are asked to do for actual jobs often differs substantially from both CS curricula [50] and interview problems [5]. As a result, performing well in technical interviews presents a distinct challenge for CS students [9].

In this study, we seek to develop a deeper understanding of how CS students prepare for technical interviews and perceive the hiring process. To that end, we conducted semi-structured interviews with 20 CS students and recent graduates with technical interview experience to answer the following research questions:

- **RQ1:** How do CS students prepare for technical interviews?
- **RQ2:** How do CS students perceive technical interviews in relation to their current coursework and their expected future job activities?

Our findings show that students primarily relied on self-directed practice using problem banks such as LeetCode[1], often feeling that the process prioritized pattern recognition over deeper problem-solving skills. They frequently collaborated with peers during preparation, which they felt to be effective. Many participants perceived a disconnect between what they learned in school, what technical

[1]https://leetcode.com

interviews assessed, and the actual tasks they were expected to perform on the job. Preparing for technical interviews demands significant time and effort, adding to students' existing academic and personal responsibilities and even leading some students to prioritize interview preparation over their own coursework. Our results highlight the need for greater alignment between academia and industry to support students in preparing effectively and equitably.

2 Background and Related Works

2.1 Technical Interviews as Hiring Practice

There are numerous techniques used to assess the technical abilities of aspiring software professionals, including take-home projects and system design interviews [58]. In this work, we focus on technical interviews, which currently represent the predominant hiring method in the tech industry [48]. This specialized form of job interview requires candidates to solve data structure and algorithm challenges using code or pseudocode, often while thinking aloud in front of interviewers [38]. Candidates applying for software development roles typically undergo multiple rounds of technical interviews as part of the hiring process [55].

Prior work has examined the perceptions of interviewees, those who undergo technical interviews in pursuit of software-related employment. Behroozi et al. analyzed comments from online forums such as Hacker News[2] and Glassdoor[3], highlighting common complaints and frustrations with the technical interview process [5, 7]. Eye-tracking studies further demonstrate that candidates often experience technical interviews as stressful and cognitively demanding [4], which may inhibit their performance [8].

Researchers have also examined insights from employers, individuals, and organizations to better understand why technical interviews are commonly required for hiring software engineers. Some sources suggest that employers perceive new hires in tech-related fields as underprepared or even unemployable [22, 39], often due to gaps between academic learning outcomes and practical skills [49]. Ford et al. found that employers value both technical and communication skills in technical interviews, highlighting a mismatch with how candidates tend to perceive the process [26]. Stepanova et al. outlined employer perspectives on hiring practices, noting a desire to better assess candidates' real-world experience [55]. Building on this work, we explore student perspectives on the misalignment between technical interview preparation, employer expectations, and the computer science curriculum.

Technical interviews have gained increased attention in computing education research and have been incorporated into classroom settings [2, 21, 37]. We extend these efforts, focusing on preparation challenges and strategies for students in relation to CS curricula and industry expectations.

2.2 Preparing for Technical Interviews

Existing research has investigated how candidates prepare for technical interviews. For example, Cui et al. analyzed user profiles on LeetCode, demonstrating that technical interviews require substantial preparation effort [17]. Since success in tech job interviews is correlated with the amount of preparation [40], the process often favors candidates who have greater time and resources available [5]. To address this disparity, researchers have explored candidate practices and educational interventions aimed at supporting more equitable technical interview preparation.

[2]https://news.ycombinator.com/
[3]https://www.glassdoor.com/index.htm

2.2.1 *Surveys/interviews of interviewees.*

Prior work leverages surveys to outline challenges and techniques for technical interview preparation. For instance, Bell et al. surveyed candidates to understand preparation methods, showing that candidates use a variety of sources, feel anxious during preparation, are often unprepared for interviews, and prioritize technical skills preparation over other aspects [9]. Kapoor et al. surveyed students to understand barriers to pursuing internships, noting that focusing on other priorities related to the undergraduate curriculum (e.g., GPA) hinders preparation [35]. Prior work also surveyed and interviewed minoritized undergraduate students, finding that they are disproportionately disadvantaged by preparation outside of class time [44] and experience anxiety during preparation [32]. We extend these works by offering qualitative insights on the CS students' preparation strategies, which they use to balance technical interview preparation with CS educational activities and job expectations.

2.2.2 *Integrating Technical Interviews as Part of the Curriculum.*

Researchers have highlighted the gap between what courses teach and what students need to be successful in technical interviews [35] and in the tech industry [50]. Educators have tried to bridge this gap by adapting their curriculum to better prepare students. For example, researchers have attempted to emphasize the importance of data structures and algorithms-related concepts that are more likely to be encountered in interviews [13, 30, 36]. Research also shows LeetCode-style exercises can benefit CS education [23, 24]. Other courses emphasize the communication aspect, which is essential in technical interviews by providing group problem-solving activities [13, 45] or including mock interviews [21, 37, 46]. For example, Cazalas et al. designed a structured course to bridge the gap between CS curriculum and industry expectations—consisting of group problem-solving activities to develop communication skills and a whiteboard interview final exam—demonstrating this approach increased student performance in learning outcomes and self-efficacy for technical interviews [13]. These activities are generally perceived as helpful and boost students' confidence [13, 37]. Yet, overhauling existing CS curricula is challenging [12]. As such, the majority of candidates report that CS courses are unhelpful for tech interviews [9]. Moreover, instructors lack resources and training to support students' interviews and job preparation [43]. In this work, we examine the (mis)alignment between students' extracurricular technical interview preparation and computing education curricula.

3 Method

In this section, we describe our methods in detail, including recruitment methods, the interview study procedure, and data analysis techniques.

3.1 Participants

For our study, we aimed to recruit current or recently-graduated university students studying CS who had experience with technical

Table 1: Information on Interviewees

Label	Gender	Background	Interviews Done
P1	Woman	Undergraduate Senior	Unknown
P2	Man	Undergraduate Senior	More than 5
P3	Man	New Grad (Employed)	More than 5
P4	Woman	New Grad (Unemployed)	2
P5	Man	Undergraduate Senior	More than 5
P6	Man	Master's Student	5
P7	Woman	Master's Student	5
P8	Man	Master's Student	More than 5
P9	Man	Master's Student	More than 5
P10	Man	PhD Student	More than 5
P11	Woman	New Grad (Employed)	3
P12	Man	Undergraduate Senior	More than 5
P13	Man	Undergraduate Junior	More than 5
P14	Woman	Undergraduate Senior	More than 5
P15	Man	Undergraduate Sophomore	2
P16	Woman	New Grad (Employed)	More than 5
P17	Man	New Grad (Employed)	5
P18	Man	Undergraduate Junior	5
P19	Man	Master's Student	4
P20	Man	Master's Student	More than 5

interviews. To that end, we recruited n=20 participants through through our university's mailing lists. Participants filled out a short demographic survey to make sure they were eligible for the study. Participants were eligible if they (1) had completed at least one technical interview; and (2) were either current university students or had graduated within the past two years.

Participant demographic information is listed in Table 1. Overall, we recruited six women and 14 men. Our participants included upper-level undergraduates (8/20), new graduates (5/20), and graduate students (7/20). Of the graduate students, the majority were Master's students (6/20). The majority of our participants had previously completed five or more technical interviews.

The study participants were interviewed during spring and fall of 2023. We did not directly collect data on which companies our participants had technical interviews with, however, our institution keeps records of the top employers per undergraduate major based on self reporting. That data is currently accessible at https://career.vt.edu/outcomes/.

3.2 Study Procedure

After verifying eligibility through the demographic surveys, we contacted participants to schedule hour-long interviews. Participants had the option to have their interview over Zoom or in person. In both cases, we recorded the audio content of the interviews.

Our semi-structured interviews covered a range of open-ended questions about the technical interview process. We asked participants how they prepared for technical interviews, including questions about the resources they used, how much time they spent, how they prepared, and whether they prepared with others or alone. We also asked participants about their general perceptions of the interview process, inquiring about their feelings both leading up to and during the interviews. We honed our interview guide through three pilot interviews, which are not reported on in this study. An outline of our interview guide can be found in Appendix A.

For their time, participants were rewarded with an electronic gift card worth 20 US dollars. The study procedure was approved by the Institutional Review Board (IRB) at the authors' university.

3.3 Data Collection and Analysis

For the first step of our analysis, we digitally transcribed the interviews and then made manual corrections by listening to the recordings and comparing them to the transcripts.

After correcting the transcriptions, we analyzed our data using reflexive thematic analysis [10, 11]. We began by conducting an open coding of our interview transcripts, inductively analyzing the text with our research questions in mind. After creating an initial codebook of open codes, two authors engaged in an affinity diagramming activity [42] to facilitate the constant comparison of existing codes [16]. Through affinity diagramming, we refined our codes and created higher-level focused codes and preliminary themes. After further review of the data, we solidified our analysis into five themes, which we present as subheadings in Section 4 and which we summarize in Table 2.

We note that because the qualitative data we collected may contain personally identifiable information, as per the policies of our university's IRB, we cannot make the data set available to the public.

4 Results

We outline our results based on two categories: preparation strategies (4.1), and the perceived relevance between the CS curriculum, technical interviews, and their desired job (4.2).

4.1 Preparing for Technical Interviews: Grinding LeetCode with Peers

4.1.1 ***Problem Solving vs. Rote Memorization.*** The primary way that the participants prepared for technical interviews was by doing practice problems. Almost all participants used LeetCode (19/20), but many also utilized other resources, such as question aggregators like Blind 75 or Neetcode 150, both of which are curated lists of LeetCode problems (11/20).

> (P5) *[The questions] I got in most of my interviews were from like either the Blind 75 list or the Neetcode 150 list.*

> (P15) *I feel like if I'd finished all of the Blind 75 questions before my first technical interview, I think I could have gotten it.*

Some participants tried to do more targeted practice specific to the company for which they are being interviewed. Participants used online resources such as Glassdoor, Blind, or levels.fyi (11/20) to find information that other candidates had shared about their experiences interviewing with a specific company.

> (P5) *I had to, like, look up the company's top questions and see what kind of pattern [there was]. And that pattern turned out to be correct when I got there.*

> (P18) *I researched company-specific questions so I can, kind of like, have an answer in mind if they ever ask it.*

> (P9) *It gives you more confidence because, you know, this is the exact question that might come your way*

Table 2: An overview of the five themes we constructed based on our interviews. For more details and supporting quotes, refer to the section for each theme.

Research Question	Theme	Description
RQ1: How do CS students prepare for technical interviews?	Problem Solving vs. Rote Memorization (4.1.1)	Participants generally prepared by doing several practice problems. Some felt this rewarded rote memorization, while others felt it gave them the ability to practice problem-solving skills.
	Practicing Together: The Role of Peer Support and Mock Interviews (4.1.2)	Participants often leveraged peer support for technical interview practice. This included both casual discussions of interview questions and also more formal mock interviews, although participants did not always have opportunities for the latter.
RQ2: How do CS students perceive technical interviews in relation to their current coursework and their expected future job activities?	(Dis)connection of Coursework to Technical Interviews (4.2.1)	Participants felt that coursework did not adequately prepare them for interviews, although data structures and algorithms courses were considered helpful. Some students were unaware that certain course material would be useful for interviews later on.
	Study-Life-Prep Balance (4.2.2)	Preparing for technical interviews can take significant time and some participants felt that candidates with more resources and fewer commitments had an advantage. We found some students prioritized interview preparation over their coursework.
	Perceived Misalignment Between Interviews and Job Expectations (4.2.3)	Many participants felt technical interviews did not accurately reflect the work that they expected to do during the actual job. Some felt that more conceptual questions or take-home assessments were more authentic.

(P3) *Sometimes, some of the companies change up the questions every so often, right? So, depends on when you took them and the questions that they asked, and sometimes not the ones that people posted previously. So, it just depends on how, I guess, lucky you got.*

While some participants simply wanted more direction about which topics to study, others were hoping to get lucky and study a problem that would be given to them in the actual interview.

Going over the curated list of problems or targeting company-specific questions reflects the desire that the topics participants practiced would appear in the actual interview—an understandable strategy given the wide range of topics they had to cover. However, several participants reflected on the tension between memorizing solutions to known problems versus practicing genuine problem-solving with unfamiliar questions.

(P7) *After practicing so many LeetCode problems, there are certain problems I have in my mind. Like, I see the question if they are directly picked from the LeetCode site. I know that these are the steps and these are the lines I have to write, which is not right. Okay. Because I'm just making up something and writing it down during the interview, which will not help me when I'm working in the industry.*

(P1) *I feel like for the people who specifically look for problems that they already know are going to be on the interview, through whatever means that they find that out through, I feel like it doesn't really test their knowledge on how to solve these problems, rather than how they just memorize how to do the solution.*

(P2) *It's not really problem solving, it really is just memorizing.*

These quotes indicate that some participants believed relying on memorized solutions could give a misleading impression of their technical skills, i.e., overestimating their problem-solving capabilities.

In contrast, other participants felt that completing a wide variety of practice problems—targeted or not—was not simply rote memorization, but instead a good way to practice problem-solving skills.

(P5) *I practiced like over like 150 problems on LeetCode, so I was able to communicate well. I guess my technical ability and problem solving, being able to pinpoint the pros and cons of certain approaches over the other, was really good.*

(P20) LeetCode style questions, they test your problem solving skills for sure.

Many participants also mentioned the value in being able to spot common patterns after completing enough practice problems—this type of "pattern-matching" could be considered a problem-solving skill when solving a similar but not-yet-seen problem.

Finally, even when some participants acknowledged an element of rote memorization in interview preparation, they noted that it can also reflect a person's persistence and willingness to prepare — traits that, while different from being immediately good at the job, may indicate a strong potential for continuous improvement once hired.

(P13) *It's the same for why schools do testing to get in. If you are willing to put in the time and effort to study for that, then you're probably willing to put in the time and effort to get better at your job. And it's not really a reflection of being good at a job. It's just a reflection of wanting to improve and putting effort into what you want to do.*

Overall, participants' reflections reveal a nuanced tension between strategic preparation and authentic assessment. While practicing known problems can increase confidence and boost performance, the students thought that it might not truly represent problem-solving capabilities. Yet, others viewed this type of preparation as a proxy for motivation and diligence—traits that are valuable in professional settings. These differing perspectives underscore the complexity of technical interviews; it may be more than a mere test of coding proficiency, raising important questions about what interviews are truly designed to measure.

4.1.2 ***Practicing Together: The Role of Peer Support and Mock Interviews.*** One significant pattern that we found from the participants was that they prepare for technical interviews with their peers (12/20), ranging from information sharing to collaborative problem solving to mock interviews.

Several participants chose to discuss interview questions and topics in a casual group setting, rather than simulating a mock interview. This communication with their friends helped them gain a new perspective when discussing problems and understand different approaches to the same issue.

> (P10) *I did my group study with a friend of mine. She was from a non CS background, so she majored in architecture, but then she moved to computer science. So her complete thought process was different from mine. [...] she would ask small things that I would just ignore, thinking, 'Oh, this is just self-explanatory' but she would keep on asking, which actually helped me think about different things.*

> (P6) *So for any problem, we give about 15 to 20 minutes to brainstorm individually. And after the initial 20 minutes, we come up with, like, what stage we are at with the problem, and we discuss. And if we are able to discuss a specific approach, then we will again, individually, try to solve the problem. And we have a certain limit of time that we spend on individual problems. And after that, if you are not able to get to the solution, we just try to refer to the Internet, like what's the approach?*

The ways that these social and interactive discussions occurred varied widely; some of them were structured, whereas others were more casual. Some people work in groups with time limits, whereas in other cases, it involves just sharing information in a group chat.

More than half of the interviewees (11/20) utilized mock interviews to prepare for the technical interview and found it helpful. In particular, they recognized the importance of the verbal communication component in technical interviews. They found that mock interviews helped them practice two-way communication, thinking aloud, and developing their thoughts through questions and answers.

> (P1) *[I would] sit down or be in a Zoom call with a friend and, kind of like a mock interview, I guess, I'm like writing the code, like really explaining through what I'm doing and asking them if there was anything that I didn't explain well, or like if they were confused by anything I was doing. That really helped me learn what I needed to talk about, like during the interview.*

> (P17) *Yeah, that [mock interview] was really helpful. Again, really built on my confidence to think properly in the moment instead of being frozen and stuck. And it also helped with communication skills, like 'Am I communicating properly? Am I being too silent? Does the other person understand what my thought process is?' I think I got the best benefit out of doing mock interviews with friends and such.*

Unfortunately, doing such mock interviews was not something that all the participants could do, even when they wished for it. Several participants (6/20) thought it would be useful, but either did not have the time or the resources to practice this way as much as they wanted to.

> (P15) *[I wish] we had like a program where they could help you with technical prep, like conduct mock interviews, [with] professors, you know, or senior students. I know a lot of clubs on campus do that, but it's very infrequent.*

These findings highlight the social aspect of interview preparation, where peer interaction, whether casual or structured, not only enhances technical understanding but also fosters the development of essential communication skills that are difficult to practice alone. On the other hand, the results also suggest that job seekers who are not in such environments, such as online learners or non-CS students, may miss out on the benefits of peer collaboration due to a lack of social capital when preparing for technical interviews.

4.2 The Tenuous Connection between Interviews, Classes, and Jobs

4.2.1 ***(Dis)connection of Coursework to Technical Interviews.*** We explicitly questioned how their coursework helped their technical interview preparation. Many participants felt that the CS curriculum did not sufficiently prepare students for technical interviews, and in some cases, did not adequately help students learn about what to expect from it There are several reasons interviewees found that the CS curriculum was not sufficient for interview preparation.

> (P14) *You could be a really good student and have straight A's, but if you've never done a single LeetCode problem, you might not do so well on your interview.*

> (P18) *I feel that the type of questions that are asked are very particular to LeetCode, HackerRank, or whatever. And there are like, designated solutions that you need to know, and that pattern recognition is just not a part of our curriculum.*

LeetCode-style problems tend not to be the focus of the CS curriculum, and this causes a gap in what is expected to perform well in a technical interview. Others also pointed out the non-typical settings in which they are asked to write code in technical interviews (e.g., non-IDE editor, thinking aloud to the interviewer). Some questioned why technical interviews did not align more closely with how they are evaluated in academic settings (e.g., "*theoretical interview questions*(P8)").

Given that many students felt the curriculum did not prepare them for technical interviews, several participants felt it would be beneficial to have additional dedicated instruction—or even a dedicated class—that focused on interview preparation.

> (P12) *I think there's like 14 main topics that are kind of large areas within these interview questions, if I recall correctly. If we had a semester-long course that went over all these, I think a lot of students would benefit from that.*

(P14) *I think it would be cool if the courses were adjusted so that you can practice solving smaller problems like the LeetCode questions, and also maybe factor in like verbalizing your solutions too.*

The above quotes show that students could see the value in dedicating class time to technical interview preparation.

While most participants felt their coursework was not sufficient preparation for technical interviews, many could still find a connection between what they learned in their courses and what they had to do during technical interviews. Half of the participants (10/20) felt that their data structures and algorithms course gave them a good foundational knowledge for the technical interview, since many LeetCode questions fall under those categories. While LeetCode-style questions may not be explicitly covered, the general concepts are.

(P6) *[Algorithms courses] give you a strong foundation. But you'll have to work on top of that to explore a different breadth of questions that is not possible to cover in a span of three to four months.*

(P12) *Yeah, I think. It's mostly probably [redacted course number] and [redacted course number], right? Because you just go over all the basic data structures and then learn about Big O, and those are the main, I think those two courses cover pretty much every data structure required for like 90% of the questions.*

Some participants, however, noted that many of the problems commonly studied for technical interviews were not covered in depth in their coursework. Others mentioned a lack of awareness at the time they took certain courses that the material would later be relevant for technical interviews. These reflections reveal a persistent mismatch between coursework and technical interview demands, underscoring students' desire for clearer connections, earlier guidance, and more explicit preparation within the curriculum itself.

4.2.2 *Study-Life-Prep Balance*.

The disconnect between the curriculum and the technical interviews underscores how additional time and effort are needed to prepare for the technical interview. Nearly all participants(19/20) found extra time necessary to figure out what is expected to excel in a technical interview, since this is not always explicitly covered during the curriculum. Since extra time needs to be taken outside of school, they shared the challenge of finding a balance between time spent at school, interview prep, and other personal obligations.

(P2) *Maybe because I just have too much going on. I lead clubs, I have like classes to do, and it's hard to find time to do a bunch of LeetCode.*

(P4) *I think that there is something to be said about people's personal backgrounds and circumstances. Some people have to work through university to support themselves. Some people don't have the luxury to get off of their classes or whatever and then sit down and do [technical interview practice] for 4 hours. Maybe they need to work part-time to pay rent, and then all they can do after that is sleep. [...] In my experience, among my peers, the ones who do best in FAANG-type situations or who are able to get jobs at those companies sometimes do come from backgrounds that have given them more time and leeway and resources to be better prepared, whether that's tutors or just more time in general to do work on code.*

Choosing how to allocate time to study for coursework, interview preparation, and other responsibilities can be difficult for interviewees. Some participants even mentioned that they prioritize technical interview prep over schoolwork, which suggests that technical interview preparation can negatively impact student learning.

(P17) *I realized studying for technical interviews is more important than my grades. So I slacked off a lot in systems [a junior-level CS course]. I don't have any regrets because I felt really prepared for interviews. My GPA was suffering to a certain point.*

(P5) *I was not able to put in any effort in one of my math classes because I was spending too much time with LeetCode, and I eventually got a C or C+ in that class because I didn't put in as much effort as I wanted to. So, I think in my opinion, I value LeetCode over my coursework and grades.*

(P11) *When I have an interview, I like to split my time up by specifically allocating a part of my day to doing interview prep only and not touching my assignments. It is really stressful when I have a deadline, but I also know that the interview is a very important part of getting a job, so I need to put that as a priority.*

These quotes reveal that technical interview preparation can harm their grades, forcing students to navigate difficult trade-offs between academic performance, career readiness, and personal obligations.

Another preparation pattern we identified is the need for participants to be ready for technical interviews at any time. Given the large number of problem sets they aim to master and the often short notice of interview scheduling, many participants described the pressure to prepare well in advance and stay consistently engaged in practice. Several noted that they had to maintain a long-term preparation routine on top of their other responsibilities, with their efforts intensifying once an interview was officially scheduled.

(P6) *So on a daily basis, I try to solve at least a couple of LeetCode problems. Typically, the time it takes for me daily is about around 2 to 3 hours. And after receiving the interview call, I'll say around 4 to 5 hours or maybe even more than that, specifically dedicated towards the interview.*

(P10) *So if I have an interview scheduled like two weeks from now, I would start working towards that. But even if I don't have any interviews, I would try to, whenever I get some time, I would try to work on it.*

Still, several participants (8/20) felt that they should practice more for technical interviews. Some fear they will forget previous problems they have practiced, or are worried that, regardless of their preparation, it will not be sufficient.

(P8) *I guess I always feel like there's more I could have done. I wouldn't say I'm lazy, but I always feel like there's like one extra percent that I could have prepared.*

(P5) *I think I should have maybe spent more time reviewing the topics I've done a lot of problems with, like reviewing the problems that I did last summer, would be much more helpful. So I'd like, remember how to do the problem.*

The uncertain nature of technical interviews contributes to this persistent anxiety of not being sufficiently ready. Even after investing significant time and effort, some participants continued to feel unprepared, suggesting that preparation is not just about mastery of content, but also about managing the anxiety and unpredictability inherent in the process.

Finally, we emphasize that the balance between interview preparation, coursework, and other personal obligations varied for each participant. For example, not all participants felt that interview preparation detracted from other areas of their lives:

> (P12) *When there are those cram times where I'm trying to like, study [for interviews] as much as possible in two weeks, obviously I'll dedicate a lot more time to that. But I wouldn't say that it's affected my personal life or academic life.*

Notably, P12 was not completely satisfied with their own interview preparation approach and felt that they ought to have a steadier work ethic.

4.2.3 ***Perceived Misalignment Between Interviews and Job Expectations.*** Some participants felt that a shortcoming of the technical interviews was that they did not provide a good representation of job expectations (8/20). They believed that day-to-day tasks were not limited to data structures and algorithms knowledge, whereas their technical interviews typically focus on those topics only.

> (P3) *I really feel like the code work you do has little to no relevance with those kinds of questions. Because, like, learning about how React state works or like how Spring Boot works or how Kafka or Mongo or whatever operate, have no relation to like, a dynamic programming question. For me, it has no correlation whatsoever.*

> (P7) *In industry, there are a lot of things that you have to take care of, like debugging or working with other teams. All those things should also be assessed [in a technical interview].*

> (P8) *And a lot of these problems don't have real environment type applications. A lot of times, you might be asked a graph question or a dynamic programming question, which one would never use. And even some of the senior software engineers in that same company would not know how to do that.*

While the above quotes are from participants who currently or previously held software engineering jobs, these attitudes were not limited to those with prior experience.

Some participants felt that LeetCode-style questions are not really a good representative of job-relevant skills and thus more appropriate to have practical questions for assessment that are specific to the position. Conceptual questions or take-home assessments were preferred by some as a better way to mirror workplace problem-solving, in addition to coping with the pressure of real-time coding while speaking. Whether these concerns stem from a student misconception or a genuine mismatch between technical interviews and actual job competencies, the result highlights the need to either reassess the alignment of hiring practices or raise student awareness about the rationale behind existing interview formats.

5 Discussion

Our findings shed light on CS students' challenges in preparing for technical interviews and their perception towards the practice in relation to their studies and the job. Our study reveals that preparation is shaped by a broad set of constraints, expectations, and trade-offs that extend far beyond simply putting in separate efforts to prepare for technical interviews. In light of our findings, we discuss the implications for CS education, as well as potential implications for the industry. Additionally, we discuss technical interventions that may also help students in preparing for technical interviews.

5.1 Implications for CS Education

In our results, we found that most participants felt they needed to put in extra effort outside of class to prepare for interviews. This aligns with prior work emphasizing the invisible labor that students must undertake to meet industry expectations outside of formal education [44]. In our interviews, we even found that some students prioritized technical interview preparation over their coursework and, by extension, their grades (see Section 4.2.2). This highlights the significance of the mismatch between the computer science curriculum and technical interview preparation. While the computer science curriculum may not be able to completely fill the gap in preparing students for technical interviews, our results suggest that efforts to narrow this gap may have significant benefits for students.

Before jumping into potential curricular improvements, it is worth asking the question: should CS instructors be expected to adapt the curriculum based on hiring practices in industry? While some may argue that the CS curriculum should not change in response to technical interview practice [19], we highlight that the gap between educational and hiring practices can adversely impact student learning by taking away time from their academic coursework (see Section 4.2.2). Expecting that students spend extra time and effort outside of their coursework to practice for technical interviews may also disadvantage students with other obligations, such as those who work jobs while attending school (see Section 4.2.2).

In addition to helping students balance their time, adapting the curriculum to support technical interview preparation can also align with the goals of educational institutions. For example, job attainment and industry preparedness, including success in hiring processes, is a long-standing goal and metric to assess institutions [1]. Additionally, many CS educators view their teaching as part of an *apprenticeship* model, where the goal of teaching is to train students to become software engineers [31]. If we are training students to become software engineers, and passing a technical interview is a vital first step towards becoming a software engineer, it seems appropriate that CS educators dedicate instruction to technical interview preparation.

So, what should instructors do to help prepare students for technical interviews? And how much time should they dedicate to interview preparation? The answer to these questions may be highly context-dependent, and likely there is no one-size-fits-all approach. For the remainder of this section, we draw upon our findings and prior work to discuss three possible interventions that instructors may use at their own discretion: adding interview-specific instruction to classes; highlighting interview-relevant material in existing classes; and finding synergistic activities to train skills that help with interviews while aligning with existing learning goals.

5.1.1 *Adding Interview-Specific Instruction.*

One option for educators to help prepare students for technical interviews is to specifically add interview preparation activities to the curriculum. Many of our participants felt this would be helpful and recognized the value in doing so (see Section 4.2.1).

One approach is to offer a course dedicated to technical interview preparation. While this can be an effective strategy [13], instructors will have to put in extra effort to design such a course, which may require input from industry partners to maintain practical relevance and authenticity [30]. Furthermore, this approach requires institutional buy-in to approve the new course offering and to determine when students should take the course, whether it fulfills degree requirements, and other logistics.

Another approach is to integrate technical interview preparation activities into existing classes [25]. This could be as simple as adding Leetcode-style exercises into assessments or active learning activities [23]. It may also be useful to include activities for practicing technical communication skills directly. While communication is important in technical interviews [26], we found that students do not always have the opportunity to explicitly practice these skills with activities like mock interviews (see Section 4.1.2). Activities like mock interviews may naturally fit into existing Data Structures and Algorithms courses that cover relevant material, and may help students to feel more confident and prepared for technical interviews [20, 36, 37].

5.1.2 *Highlighting Interview-Relevant Topics.*

One potential low-effort intervention for instructors is to let students know early on what topics they might expect to see during technical interviews and highlight these relevant topics throughout the course. While the students we interviewed felt that the material covered in their data structures and algorithms (DSA) courses was helpful for technical interviews, some noted that they did not know this at the time when they were learning the material (see Section 4.2.1). Others have also noted that students in DSA courses do not necessarily know what to expect from technical interviews [36].

Proactively sharing information about technical interviews before covering relevant topics might not only help students better understand technical interviews, but it may also help motivate them to learn the material. Explaining to students the connection between what they are learning and their career goals (e.g., passing an interview and getting a job) can be an important factor in student motivation [33]. While students might not readily see how traversing a binary search tree will be useful in their future day-to-day job as a programmer, they might be more easily convinced that this information will prove useful when demonstrating their technical knowledge and abilities during a job interview.

5.1.3 *Finding Synergistic Activities.*

Being good at technical interviews is not its own specific skill; rather, there are a variety of skills needed to succeed in technical interviews [26], such as deciding which algorithms to use under time pressure, thinking aloud about problem-solving strategies, and communicating effectively with interviewers to make sure they understand the approach. Thus, instead of specifically training students for technical interviews, CS instructors may be able to find synergistic activities that help students develop useful skills for technical interviews while also advancing existing learning objectives.

Because communication is an important skill for technical interviews [26], instructors might try to foster technical communication skills by incorporating more group problem-solving activities in the classroom or as part of the coursework [20, 37]. In our interviews, students felt that studying interview problems in a group was effective for gaining new perspectives and understanding (see Section 4.1.2), and prior work suggests that candidates who prepare with others feel significantly more prepared than those who do not [9]. In general, learning theory research suggests that more interactive activities where learners communicate with each other are particularly effective for learning [14]. We suspect that activities involving technical communication about programming and problem-solving with code would transfer useful skills for technical interviews, even when these activities are not explicitly designed with technical interviews in mind. Facilitating social learning through in-person interactions may also help students build social capital they can later leverage during interview preparation—for example, by forming peer groups to practice together.

Another pedagogical strategy that may incidentally help students with technical interviews is *live coding*, a commonly used lecture technique where instructors write code in front of the class during their lecture, thinking aloud to share their coding process with students [47, 51, 57]. Live coding can give students an explicit example of thinking aloud while coding, a skill that our participants found valuable in our interviews (see Section 4.1.2), potentially reducing anxiety in traditional interview settings [6, 8]. Spectatorship, the process of observing others complete a task [52], can enhance learning by helping students understand the interview process from the interviewer's perspective. In addition to having students watch the instructor, some instructors also advocate for directly involving students in live coding, for example, by having them actively guide the instructor's coding process [54] or by having students themselves live code in front of the class [27, 28]. Modeling the problem-solving process by thinking aloud and having students articulate and communicate about process are both important tools for teaching process-oriented skills [15], and thus may help students become better programmers while also helping them improve at technical interviews.

5.2 Implications for Industry

Our study raises critical questions about what technical interviews are truly measuring: problem-solving skills, verbal communication, technical proficiency, conscientiousness, or all of the above. Part of the complexity originates from the performative nature of interviews, which require effective problem-solving and technical communication with artificial time constraints and under conditions that do not necessarily reflect normal working conditions [8, 26]. Another issue is the perceived lack of alignment between skills that are useful for technical interviews versus skills that are useful on the job, a concern shared by many of our participants (see Section 4.2.3) and also commonly expressed on tech-focused forums [5]. Thus, one possible improvement to the interviewing process is for tech companies to explore alternatives to typical DSA-focused technical interviews, as some companies have started to do [58].

We posit that bringing technical interviews more in line with job expectations has two main benefits. The first is that it has the potential to make technical interview performance more highly correlated with future job performance. The second is that by testing candidates for skills that will be useful on the job, companies may in turn encourage students to spend their time and energy developing these sought-after skills. Our findings suggest that many students may be motivated to place a significant amount of effort into interview preparation outside of their coursework (see Section 4.2.2). This extra effort might be harnessed to develop experience and knowledge in skills more directly applicable to software engineering, such as being able to handle the complexity of working with large code bases [41, 53]. Future work may explore how a wider variety of software engineering skills might be effectively assessed in technical interviews.

5.3 Supporting Technical Interview Practice

Outside of curricular interventions, we might also try to help students prepare for technical interviews by offering systems that facilitate interview practice. One issue our participants had when preparing for interviews is that they could not always find people to study or conduct mock interviews with (see Section 4.1.2). One solution is to build a system that facilitates pairing candidates for mock interviews. Another solution is to provide on-demand technical interview practice through AI assistance [18]. While interacting with an AI-interviewer might not be as ecologically valid as interacting with a human, students may feel vulnerable practicing their problem-solving skills in front of others and might value the privacy and anonymity of interacting with AI to do so [18, 29].

Another potential approach is to build systems that encourage students to reflect on their problem-solving performance, even when not part of a mock interview. For example, researchers have found that systems that allow students to watch replays of their coding process can lead to positive self-regulation behaviors [59]. Additionally, systems that facilitate students to asynchronously compare their own problem-solving approach with that of others can be useful in helping students reflect on and refine their approach [15]. These types of systems may complement the social approaches to preparation we saw with our participants, or can provide a flexible alternative when students cannot find peers to collaborate with.

6 Limitations and Future Work

While our study offers valuable insights into the challenges CS students encounter when preparing for technical interviews, it may not fully account for the relative significance or prevalence of each challenge due to the qualitative nature of the study. In addition, our sample is limited to students and recent graduates from a single university, which may constrain the generalizability of our findings. Students at other institutions, particularly those with different curricular structures, geographical locations, levels of institutional support, or demographic compositions, may encounter different challenges or experience similar issues to varying degrees. As such, additional research across diverse institutional contexts is needed to validate and expand upon the patterns identified in this study. We also only collect student perspectives, and future efforts focusing on instructors and employers can provide additional insights.

Additionally, our study captures a single snapshot in time, while the landscape of software engineering jobs and interview practices continues to evolve. As such, our results may reflect current trends, such as a relatively difficult job market for entry-level software engineers compared to the past [3]. We note that this limitation is inherent to any study of technical interviews.

Finally, future work is needed to better understand the interventions discussed in Section 5.1 and 5.3. While we have some evidence that curricular interventions can lead to increased student confidence in technical interviews [37], we do not know whether these interventions actually work to increase student technical interview performance. In addition, future work can examine to what extent the active learning and live coding approaches discussed in Section 5.1.3 can benefit students preparing for technical interviews, or if these techniques might be adapted for more targeted benefits. Similarly, future work may examine how and to what extent systems such as those discussed in Section 5.3 can help students prepare for technical interviews even when they cannot find peers to practice with.

7 Conclusion

In this study, we aimed to better understand how CS students prepare for technical interviews and how they perceive these interviews in relation to the CS curriculum and industry job expectations. To that end, we conducted 20 semi-structured interviews with current or recently-graduated CS students who have had recent experience with technical interviews.

We analyzed the interviews using Thematic Analysis and constructed five themes from the data. We found that participants relied on platforms like LeetCode and prepared by repeated practice with coding problems, even though some felt that this prioritized blind pattern-matching over practical problem-solving skills. We found participants often leveraged peer support for technical interview practice, including casual discussions of interview problems and also more formal mock interviews. Many participants reported that their coursework was not sufficient to prepare them for technical interviews, leading some to prioritize interview preparation over their academic responsibilities. Many participants felt that technical interviews did not reflect their job expectations, and some expressed a desire for interview formats that better reflect job-relevant skills.

We discuss our findings and offer insights into how academic institutions and employers might better support software engineering job candidates. While our study emphasizes in-depth qualitative insights from a single institution, future multi-institutional studies may wish to further explore the prevalence of the interview preparation techniques, challenges, and attitudes we uncovered in this work.

Acknowledgments

This work was supported by a Google Award for Inclusion Research.

References

[1] 2024. Colleges with Strong Computer Science Job Placement. *College Vine* (2024). https://www.collegevine.com/faq/159543/colleges-with-strong-computer-science-job-placement.

[2] Giulia Alberini and Elena Bai. 2025. Implementation of Technical Interviews as an Alternative Assessment in a Large Introductory CS Course. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. New York, NY, USA, 1361–1362.

[3] Jake Angelo. 2025. US tech job postings remain below pre-pandemic levels. https://www.staffingindustry.com/news/global-daily-news/us-tech-job-postings-remain-below-pre-pandemic-levels. Staffing Industry Analysts.

[4] Mahnaz Behroozi, Alison Lui, Ian Moore, Denae Ford, and Chris Parnin. 2018. 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* (Gothenburg, Sweden). IEEE, 93–96.

[5] Mahnaz Behroozi, Chris Parnin, and Titus Barik. 2019. Hiring is Broken: What Do Developers Say About Technical Interviews?. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Memphis, TN, USA, 1–9. https://doi.org/10.1109/VLHCC.2019.8818836

[6] Mahnaz Behroozi, Chris Parnin, and Chris Brown. 2022. 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* (Singapore, Singapore) *(ESEC/FSE 2022)*. New York, NY, USA, 294–305.

[7] Mahnaz Behroozi, Shivani Shirolkar, Titus Barik, and Chris Parnin. 2020. Debugging hiring: What went right and what went wrong in the technical interview process. In *International Conference on Software Engineering: Software Engineering in Society (ICSE SEIS)*. https://doi.org/10.1145/3377815.3381372

[8] Mahnaz Behroozi, Shivani Shirolkar, Titus Barik, and Chris Parnin. 2020. 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*. 481–492.

[9] Brian Bell, Teresa Thomas, Sang Won Lee, and Chris Brown. 2025. How do Software Engineering Candidates Prepare for Technical Interviews?. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering* (Clarion Hotel Trondheim, Trondheim, Norway) *(FSE Companion '25)*. Association for Computing Machinery, New York, NY, USA, 883–894. https://doi.org/10.1145/3696630.3727245

[10] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[11] Virginia Braun, Victoria Clarke, Nikki Hayfield, Louise Davey, and Elizabeth Jenkinson. 2022. Doing Reflexive Thematic Analysis. In *Supporting Research in Counselling and Psychotherapy : Qualitative, Quantitative, and Mixed Methods Research*, Sofie Bager-Charleson and Alistair McBeath (Eds.). Springer International Publishing, Cham, 19–38. https://doi.org/10.1007/978-3-031-13942-0_2

[12] João MP Cardoso. 2005. New challenges in computer science education. *ACM SIGCSE Bulletin* 37, 3 (2005), 203–207.

[13] Jonathan Cazalas, Christian Roberson, and Zeeshan Furqan. 2024. From Degree to Developer: The Creation and Evolution of a CS Course Designed to Bridge the Academia-Industry Gap. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. New York, NY, USA, 186–192.

[14] Michelene T. H. Chi and Ruth Wylie. 2014. The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes. *Educational Psychologist* 49, 4 (Oct. 2014), 219–243. https://doi.org/10.1080/00461520.2014.965823

[15] Allan Collins, John Seely Brown, Ann Holum, et al. 1991. Cognitive apprenticeship: Making thinking visible. *American educator* 15, 3 (1991), 6–11.

[16] Juliet Corbin and Anselm Strauss. 2014. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

[17] Jialin Cui, Runqiu Zhang, Fangtong Zhou, Ruochi Li, Yang Song, and Ed Gehringer. 2024. How Much Effort Do You Need to Expend on a Technical Interview? A Study of LeetCode Problem Solving Statistics. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*. 1–10. https://doi.org/10.1109/CSEET62301.2024.10663022

[18] Taufiq Daryanto, Sophia Stil, Xiaohan Ding, Daniel Manesh, Sang Won Lee, Tim Lee, Stephanie Lunn, Sarah Rodriguez, Chris Brown, and Eugenia Rho. 2025. Designing Conversational AI to Support Think-Aloud Practice in Technical Interview Preparation for CS Students. arXiv:2507.14418 [cs.HC] https://arxiv.org/abs/2507.14418

[19] Erik Dietrich. 2024. Deploying Guerrilla Tactics to Combat Stupid Tech Interviews. *DaedTech* (2024). .

[20] Edward Dillon, Abigail Dina, Mariah McMichael, Theodore Wimberly Jr, Lauren Brown, and Krystal L Williams. 2023. Exposing Early CS Majors to Technical Interview Practices in the Form of Group-Based Whiteboard Problem Solving Activities. In *2023 ASEE Annual Conference & Exposition*.

[21] Edward Dillon, Briana Williams, Ayomide Ajayi, Zipporah Bright, Quinlan Kimble-Brown, Chauncey Rogers, Myles Lewis, Joseph Esema, Ben Clinkscale, and Krystal L. Williams. 2021. Exposing Early CS Majors to Coding Interview Practices: An HBCU Case Study. In *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*.

[22] James P Downey, Mark E McMurtrey, and Steven M Zeltmann. 2008. Mapping the MIS curriculum based on critical skills of new graduates: An empirical examination of IT professionals. *Journal of Information Systems Education* 19, 3 (2008), 351–363.

[23] Stephen H. Edwards and Krishnan Panamalai Murali. 2017. CodeWorkout: Short Programming Exercises with Built-in Data Collection. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. NY, USA, 188–193.

[24] Stephen H. Edwards, Krishnan P. Murali, and Ayaan M. Kazerouni. 2019. The Relationship Between Voluntary Practice of Short Programming Exercises and Exam Performance. In *Proceedings of the ACM Conference on Global Computing Education (CompEd '19)*. NY, USA, 113–119.

[25] Rachel Field, Edward Dillon, and Steven J Fuller. 2023. Surveying the Importance of Integrating Technical Interviews into Computer Science Curriculums and Increasing Awareness in the Academy. In *2023 ASEE Annual Conference & Exposition*.

[26] Denae Ford, Titus Barik, Leslie Rand-Pickett, and Chris Parnin. 2017. 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)*. 43–48. https://doi.org/10.1109/CHASE.2017.8

[27] Alessio Gaspar and Sarah Langevin. [n. d.]. Active learning in introductory programming courses through Student-led "live coding" and test-driven pair programming. ([n. d.]).

[28] Alessio Gaspar and Sarah Langevin. 2007. Restoring "coding with intention" in introductory programming courses. In *Proceedings of the 8th ACM SIGITE conference on Information technology education (SIGITE '07)*. Association for Computing Machinery, New York, NY, USA, 91–98. https://doi.org/10.1145/1324302.1324323

[29] Katy Ilonka Gero, Tao Long, and Lydia B Chilton. 2023. Social Dynamics of AI Support in Creative Writing. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 245, 15 pages. https://doi.org/10.1145/3544548.3580782

[30] Kinnis Gosha, Vinesh Kannan, Lee Morgan, and Earl W Huff Jr. 2019. Strategic partnerships to enhance data structures and algorithms instruction at HBCUs. In *Proceedings of the 2019 ACM Southeast Conference*. 194–197.

[31] Mark Guzdial. 2015. Learner-Centered Design of Computing Education: Research on Computing for Everyone. *Synthesis Lectures on Human-Centered Informatics* 8, 6 (Nov. 2015), 1–165. https://doi.org/10.2200/S00684ED1V01Y201511HCI033 Publisher: Morgan & Claypool Publishers.

[32] Phillip Hall Jr. and Kinnis Gosha. 2018. 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*. ACM, Buffalo-Niagara Falls NY USA, 64–69.

[33] Brett D. Jones. 2009. Motivating Students to Engage in Learning: The MUSIC Model of Academic Motivation. *International Journal of Teaching and Learning in Higher Education* 21, 2 (2009), 272–285. https://eric.ed.gov/?id=EJ899315 Publisher: International Society for Exploring Teaching and Learning ERIC Number: EJ899315.

[34] Tobias Kaatz. 2014. Hiring in the Software Industry. *IEEE Software* 31, 6 (2014), 96–96. https://doi.org/10.1109/MS.2014.140

[35] Amanpreet Kapoor and Christina Gardner-McCune. 2020. Barriers to Securing Industry Internships in Computing. In *Proceedings of the Twenty-Second Australasian Computing Education Conference (ACE'20)*. NY, USA, 142–151.

[36] Amanpreet Kapoor and Christina Gardner-McCune. 2021. 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*. 633–634.

[37] Amanpreet Kapoor, Sajani Panchal, and Christina Gardner-McCune. 2023. Implementation and Evaluation of Technical Interview Preparation Activities in a Data Structures and Algorithms Course. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023)*. New York, NY, USA, 882–888. https://doi.org/10.1145/3545945.3569755

[38] Gayle Laakmann. 2015. *Cracking the Coding Interview*. CareerCup.

[39] Choong Kwon Lee and Hyo-Joo Han. 2008. Analysis of skills requirement for entry-level programmer/analysts in Fortune 500 corporations. *Journal of Information Systems Education* 19, 1 (2008).

[40] Aline Lerner. 2024. The technical interview practice gap, and how it keeps underrepresented groups out of software engineering. *interviewing.io* (2024). https://interviewing.io/blog/technical-interview-practice-gap.

[41] Paul Luo Li, Amy J. Ko, and Jiamin Zhu. 2015. What Makes a Great Software Engineer?. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 700–710. https://doi.org/10.1109/ICSE.2015.335

[42] Andrés Lucero. 2015. Using Affinity Diagrams to Evaluate Interactive Prototypes. In *Human-Computer Interaction – INTERACT 2015*, Julio Abascal, Simone Barbosa, Mirko Fetter, Tom Gross, Philippe Palanque, and Marco Winckler (Eds.). Springer International Publishing, Cham, 231–248.

[43] Stephanie Jill Lunn, Edward Dillon, and Zubayer Ahmed Sadid. 2024. Educational Expertise: Faculty Insights on Preparing Computing Students to Navigate Technical Interviews. In *2024 ASEE Annual Conference & Exposition*.

[44] Stephanie Jill Lunn, Ellen Zerbe, and Monique Ross. 2024. You're Hired! A Phenomenographic Study of Undergraduate Students' Pathways to Job Attainment in Computing. *ACM Trans. Comput. Educ.* 24, 1 (Jan. 2024), 7:1–7:29.

[45] Bonnie MacKellar. 2012. A case study of group communication patterns in a large project software engineering course. In *2012 IEEE 25th Conference on Software Engineering Education and Training*. IEEE, 134–138.

[46] Kaylah Mackroy, Whitney Nelson, Talitha Washington, and Kinnis Gosha. 2023. Virtual Post Baccalaureate Technical Interview Develpoment for Black Software Engineers. In *2023 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. 177–181.

[47] Daniel Manesh, Tong Wu, Yan Chen, and Sang Won Lee. 2025. Understanding and Improving Student Note-Taking in Live Coding Lectures. In *Proceedings of the 2025 ACM Conference on International Computing Education Research V.1 (ICER '25)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3702652.3744224

[48] Gayle Laakmann McDowell. 2019. *Cracking the Coding Interview: 189 Programming Questions and Solutions*. CareerCup.

[49] Craig S Miller and Lucia Dettori. 2008. Employers' perspectives on it learning outcomes. In *Proceedings of the 9th ACM SIGITE conference on Information technology education*. 213–218.

[50] Damla Oguz and Kaya Oguz. 2019. Perspectives on the gap between the software industry and the software engineering education. *Ieee Access* 7 (2019), 117527–117543.

[51] Adalbert Gerald Soosai Raj, Pan Gu, Eda Zhang, Arokia Xavier Annie R, Jim Williams, Richard Halverson, and Jignesh M. Patel. 2020. Live-coding vs Static Code Examples: Which is better with respect to Student Learning and Cognitive Load?. In *Proceedings of the Twenty-Second Australasian Computing Education Conference (ACE'20)*. New York, NY, USA, 152–159.

[52] Leah F Rosenbaum. 2024. Spotlighting spectatorship: elevating observation-based learning in the design and evaluation of body-scale learning environments. *Educational technology research and development* 72, 4 (2024), 2133–2157.

[53] Anshul Shah, Thomas Rexin, Anya Chernova, Gonzalo Allen-Perez, William G. Griswold, and Adalbert Gerald Soosai Raj. 2025. Needles in a Haystack: Student Struggles with Working on Large Code Bases. In *Proceedings of the 2025 ACM Conference on International Computing Education Research V.1 (ICER '25)*. Association for Computing Machinery, New York, NY, USA, 27–40. https://doi.org/10.1145/3702652.3744218

[54] Amy Shannon and Valerie Summet. 2015. Live coding in introductory computer science courses. *Journal of Computing Sciences in Colleges* 31, 2 (Dec. 2015), 158–164.

[55] Anna Stepanova, Alexis Weaver, Joanna Lahey, Gerianne Alexander, and Tracy Hammond. 2021. Hiring CS Graduates: What We Learned from Employers. *ACM Trans. Comput. Educ.* 22, 1 (Oct. 2021), 5:1–5:20. https://doi.org/10.1145/3474623

[56] Glenn Stovall. 2021. Whiteboard Interview Guide: The Good, The Bad, and The Ugly. *CoderPad* (2021). https://coderpad.io/blog/interviewing/whiteboard-interview-guide/.

[57] Xiaotian Su and April Yi Wang. 2025. The Stress of Improvisation: Instructors' Perspectives on Live Coding in Programming Classes. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25)*. Association for Computing Machinery, New York, NY, USA, Article 525, 6 pages. https://doi.org/10.1145/3706599.3719993

[58] Lauren Tan. [n. d.]. Hiring Without Whiteboards. https://github.com/poteto/hiring-without-whiteboards.

[59] Benjamin Xie, Jared Ordona Lim, Paul K.D. Pham, Min Li, and Amy J. Ko. 2023. Developing Novice Programmers' Self-Regulation Skills with Code Replays. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1 (ICER '23, Vol. 1)*. Association for Computing Machinery, New York, NY, USA, 298–313. https://doi.org/10.1145/3568813.3600127

A Interview Guide

We used the following questions to guide our semi-structured interviews:

- How early in the process do you prepare for technical interviews? (e.g., as soon as one is scheduled vs. continuously preparing)
- About how much time do you dedicate to preparing for technical interviews? (per session? over how long?)
- What do you do to prepare for technical interviews?
- What kinds of resources do you use and how do you use them?
- How much information have you received from the companies about the technical interviews?
- Do you prepare differently based on the company?
- Have you ever worked with your peers or friends to prepare for interviews? If so, how was that experience?
- How do you practice the social aspect of interviews, such as thinking aloud?
- How would your ideal preparation process look? Would it be different from what you currently do?
- How do you know if you are adequately prepared?
- Can you share one interview experience that went well and one that did not go well and discuss why?
- What do you think makes a good interviewer?
- What is your opinion on whiteboarding interviews? Would you prefer a different approach?
- How does studying for technical interviews fit in to your academic and general daily life?
- What has been the most difficult thing you have faced during your study process?
- How has your coursework helped you prepare for interviews, if at all? What classes have been useful?