How do Software Engineering Candidates Prepare for Technical Interviews?

Brian Bell
Virginia Tech
bbell97@vt.edu

Teresa Thomas
Virginia Tech
teresa2020@vt.edu

Sang Won Lee
Virginia Tech
sangwonlee@vt.edu

Chris Brown
Virginia Tech
dcbrown@vt.edu

Abstract

To obtain employment, aspiring software engineers must complete *technical interviews*—a hiring process which involves candidates writing code while communicating to an audience. However, the complexities of tech interviews are difficult to prepare for and seldom faced in computing curricula. To this end, we seek to understand how candidates prepare for technical interviews, investigating the effects of preparation methods and the role of education. We distributed a survey to candidates (n = 131) actively preparing for technical interviews. Our results suggest candidates rarely train in authentic settings and courses fail to support preparation efforts—leading to stress and unpreparedness. Based on our findings, we provide implications for stakeholders to enhance tech interview preparation for candidates pursuing software engineering roles.

CCS Concepts

• **Social and professional topics** → **Computing profession**.

Keywords

technical interviews, candidate preparation, computing education

ACM Reference Format:
Brian Bell, Teresa Thomas, Sang Won Lee, and Chris Brown. 2025. How do Software Engineering Candidates Prepare for Technical Interviews?. In *33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25), June 23–28, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3696630.3727245

Prelude

Consider the *Two Sum* problem depicted in Figure 1, an "easy" challenge on LeetCode [6]—an online platform for technical interview preparation. Take a moment to try and solve this problem—Were you able to come up with a solution? Does it pass the provided test case? What are the time and space complexity? Can you verbally describe how you came up with your solution? Could you do this while being watched by an interviewer?—According to the LeetCode community, this problem was used in recent tech interviews to hire software engineers at companies such as Google, Meta, Amazon, and LinkedIn.[1]

[1] https://leetcode.com/discuss/interview-experience?query=twosum

FSE Companion '25, June 23–28, 2025, Trondheim, Norway

ACM ISBN 979-8-4007-1276-0/2025/06
https://doi.org/10.1145/3696630.3727245

1 Introduction

Software engineers are primarily hired through an evaluation process known as *technical interviews*. Traditional technical interviews, or tech interviews, consist of candidates writing code to solve a programming challenge related to data structures and algorithms–typically in a non-programming environment such as a whiteboard or online text editor (*i.e.,* Google Docs or CollabEdit)–while performing a *think-aloud* to verbally communicate their problem-solving strategy and thought process to observing interviewer(s) [78, 93]. This process is widely used by companies to evaluate the proficiency of candidates pursuing software engineering (SE) positions, assessing the technical and soft skills needed to work with a team to develop complex and high-quality software systems [58].

However, the experiences of candidates provide a different perspective of current tech hiring practices. Technical interviews invoke negative perceptions and frustrations from developers, who criticize this process for its data structure-focused nature and irrelevance to real-world SE work, among other complaints [20, 22]. Further, companies that do not utilize traditional tech interviews in their hiring processes are praised for hiring without whiteboards [94]. Research also shows technical interviews increase anxiety and lead to worse performances for candidates [23] and a "leaky pipeline" in which qualified candidates are overlooked or burn out in tech hiring processes [90].

Technical interviews are a cognitively and socially demanding activity—assessing candidates' problem-solving ability, technical knowledge, computational thinking, and communication—making them difficult to prepare for. Recent studies by interviewing.io[2] suggest 54% of candidates pass interviews [67] and only 20% perform consistently between interviews [66]. Further, aspiring software engineers rarely face authentic technical interview settings—writing code with simultaneous think-aloud in front of observers—in Computer Science (CS) education or in practice, increasing the burden on individuals to prepare [24]. Yet, a notable complaint from developers is the amount of preparation time and effort needed to be competitive [20, 39]—with numerous resources recommending candidates spend multiple hours per day practicing coding problems to obtain offers [9, 53, 82]. Yet, this leads to increased anxiety, particularly for underrepresented job seekers [52, 73], and bias favoring candidates with more time and resources to prepare [20]. Moreover, candidates receive no feedback after completing technical interviews, only a job offer or not, making it difficult to improve for future interviews without any skills assessment of past performances [22]. In addition to preparation difficulties, candidates do not learn interview concepts in CS curricula and instructors face challenges teaching tech interview skills to help students succeed [74].

[2] https://interviewing.io, a tech interview prep website

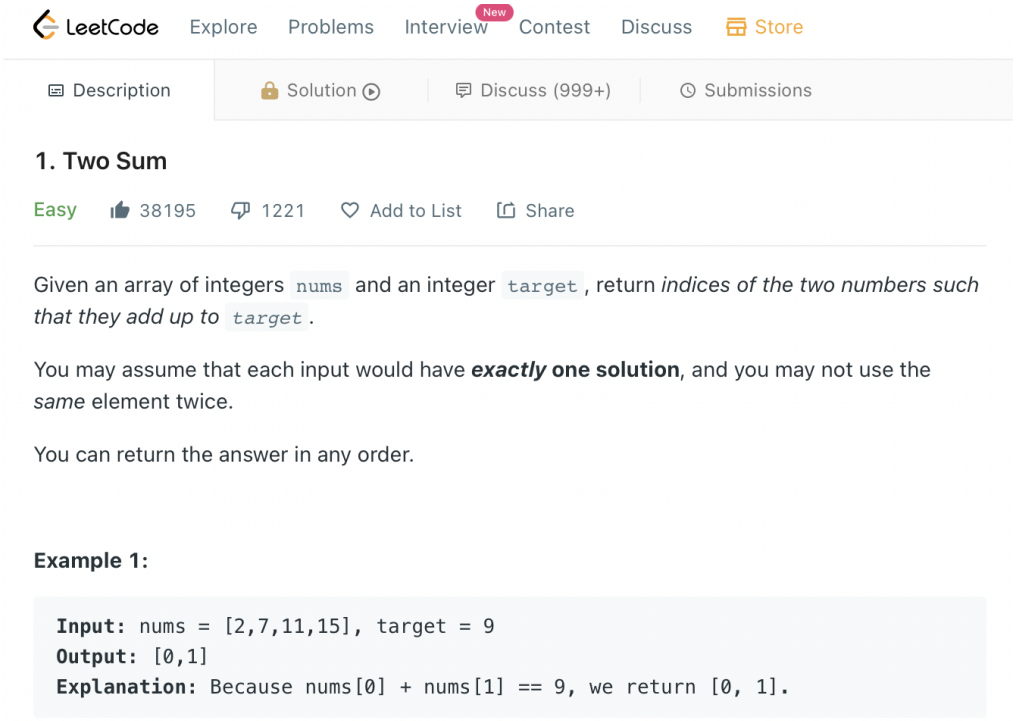

Figure 1: Two Sum technical interview problem, as portrayed by https://leetcode.com/problems/two-sum/

In this work, we aim to investigate how candidates *prepare* for the complexities of technical interviews and examine the role of computing education in preparation efforts. To this end, we aim to answer the following research questions (RQs):

- **RQ1** How do candidates prepare for technical interviews?
- **RQ2** How do candidates perceive the effects of technical interview preparation?
- **RQ3** How do candidates perceive the impacts of SE education on preparing for technical interviews?

To answer these questions, we distributed an online survey to collect data on candidates' ($n = 131$) preparation resources and methods, the perceived effectiveness of their approaches, and the adequacy of their computing education to support preparation efforts. We found candidates use a variety of preparation resources—yet rarely practice in authentic environments, leading to a lack of preparedness and anxiety for technical interviews. Further, participants suggest computing education is unsupportive for tech interview preparation, expressing a desire for increased support in academic settings. Based on our results, we discuss opportunities to enhance technical interview preparation. Particularly, we provide guidelines for candidates, employers, interview preparation resources, and higher education—focusing on the learning and teaching of technical interview concepts—to enhance candidates' preparedness for complex technical interview environments.

2 Motivating Example

Referring back to the "Two Sum" problem presented in the Prelude, we provide a brief explanation to demonstrate the difficulties of technical interview preparation.

First, candidates must implement a solution to the problem in a selected programming language or pseudocode. The brute force algorithm involves looping through each element x to find if there is a second element which meets the case of equaling *target* - x. This solution is inefficient with an $O(n^2)$ time complexity, however only uses $O(n)$ space complexity. Thus, candidates must consider more complex data structures and algorithms to provide a more efficient solution.

Another solution is "`Two-pass Hash Table`" [44]. This algorithm uses a hash table data structure to reduce the program runtime by mapping array elements to their index. Candidates must recognize this algorithm decreases the time complexity of the solution to $O(n)$ and maintains $O(n)$ space complexity. These are just two of the many solutions that exist for this problem—and Two Sum is one of the innumerable number of problems that could be asked during interviews. In addition to a valid technical solution, candidates must also be prepared to provide a clear verbal rationale describing their approach, which is critical in hiring decisions [47], in front of one or more interviewers assessing their performance.

While advanced data structures such as hash tables are commonly taught in CS education, they are complex for students to understand [80] and infrequently taught across institutions [92]—causing students to lack educational training on how to apply these constructs to coding challenges and communicate effectively about them. Moreover, despite being a critical part of software development [68], training in communication and soft skills are often overlooked computing curricula and SE courses [50]. Consequently, candidates must spend a considerable amount of individual time and effort preparing for technical interviews [20, 39].

3 Related Work

This project extends prior work investigating technical interviews. Research shows tech interviews are problematic, with developers finding current processes irrelevant, anxiety-inducing, and biased [20]. Behroozi and colleagues analyzed comments posted by software developers on Glassdoor [3], an online platform for job reviews, and found additional concerns, such as a bias towards candidates with more time and resources to prepare [22]. Ford *et al.* found mismatches between candidate expectations for tech interviews and hiring manager assessment criteria for hiring decisions [47]. Further, Behroozi *et al.* used head-mounted eye trackers to show increased levels of stress in candidates during technical interviews, resulting in a negative impact on candidates' performance–particularly those identifying as women [19, 23]. To reduce stress during interviews, asynchronous interview approaches have been proposed and shown to enhance the technical and communication ability of candidates [21]. Finally, Lunn *et al.* show negative tech interviews adversely impact students' computing identity [72].

Research has also investigated technical interview preparation. Behroozi *et al.* show interview preparation is frustrating for software practitioners [20]. Cui *et al.* observed preparation techniques for LeetCode users, highlighting correlations between LeetCode ratings and job offers at top tech companies [39]—demonstrating the substantial prep time and effort needed to succeed in tech interviews. Interview preparation has also been explored as a barrier to inclusive computing workforces. For instance, Hall and Gosha outline anxieties in technical interview preparation faced by CS students from Historically Black Colleges and Universities (HBCUs)—who often have less resources than CS departments at other institutions [52]. Similarly, Lunn *et al.* show racial and gender minority students are more likely to have cultural experiences that inhibit their ability to prepare [73]. Our study extends this work by examining how candidates prepare for technical interviews—contributing insights on the resources and authenticity of training processes used, their perceived effects on preparedness, and the impact of SE education curricula on technical interview preparation for candidates pursuing SE positions in the tech industry.

4 Methodology

The goal of this research is to investigate study patterns candidates use to prepare for technical interviews. We acknowledge other types of SE hiring evaluations exist [95], such as take-home coding assessments or interviews focused solely on behavioral qualities without invoking technical skills. However, the scope of our study targets preparation for *whiteboard-styled technical interviews*—or tech interviews that involve candidates writing code to solve data structures and algorithms-based problems in a non-programming environment while thinking aloud in the presence of one or more interviewers [93].

4.1 Online Surveys

We deployed online surveys to collect data from participants actively preparing for tech interviews to obtain SE-related positions. Our study was approved by our institutional review board (IRB).

4.1.1 Participant Recruitment. We used purposive sampling to recruit participants from two main sources. The survey was distributed on department listservs at the authors' institution, targeting students preparing for tech interviews to obtain internship and full-time positions. We also collaborated with Exponent [1] and Pramp [8] to share our survey, allowing us to reach a broader sample. Exponent and Pramp are online platforms to help candidates prepare for technical interviews, providing a variety of features such as coding problems, courses, peer mentoring, and expert coaching. Surveys were placed on the homepage of the respective websites to recruit participants.

4.1.2 Survey Design. We distributed three versions of the survey each to our institution, Exponent, and Pramp. The surveys collected data on participants' demographic background, general interviewing experience, preparation techniques and challenges, and the role of university education. The three surveys were identical, except the Exponent and Pramp versions contained an additional section specifically collecting feedback on those systems. The responses to these questions were shared directly with our industry partners, and are not included in our analysis as they do not relate to our research goals. The survey included a variety of short-answer, open-ended, and Likert scale questions to describe their technical interview training methods. The questions were formed based on our RQs and insights from our industry partners, and refined through reviews from experts well-versed in empirical SE research and tech interviews as well as a pilot study with two representative participants from our institution, who are excluded from our data. The surveys are available online.[3]

4.2 Data Analysis

We used a mixed methods approach to analyze the survey data. For Likert scale questions, we leverage the Mann-Whitney U test to analyze responses across groups based on participants' demographic background and preparation practices [16]. For continuous data, we divide participants into two groups (high/low) based on the median value [54]. For open-ended questions, we used an open coding approach to derive themes from participant responses [27]. Two researchers independently analyzed and categorized responses, then came together to discuss their findings and resolve disagreements. After this process, the researchers finalized a set of themes based on participant responses. We did not obtain IRB approval to share survey responses, however this data can be made available upon request after receiving the necessary permissions.

4.3 Participants

We received a total of 131 responses from a diverse sample of participants actively preparing for technical interviews. For each survey, we received 93 responses from Exponent users, 34 from Pramp users, and four from our university. 104 participants (79.4%) identified as male while 27 (20.6%) identified as female. In terms of race/ethnicity, we found 64 participants (48.9%) identified as Asian, 34 (25.9%) as White, 17 (13.0%) as Black or African American, 15 (11.5%) as Hispanic/Latino, five (3.8%) as American Indian or Alaskan Native, and 11 (8.4%) as two or more races.

[3] https://github.com/code-world-no-blanket/TechInterviewPrep

We found our study population represents a mixture of candidates pursuing SE careers, including students completing technical interviews to obtain their first job or an internship as well as experienced software engineers in employment transition. 35 participants (26.7%) self-identified as students–representing freshmen (40%, $n = 14$), juniors (22.9%, $n = 8$), seniors (31.4%, $n = 11$), Masters (20%, $n = 7$), and PhD (5.7%, $n = 2$) students at a wide range of global universities. Most participants ($n = 96$, 73.3%) identified as SE professionals in industry, representing a variety of companies. Overall, participants averaged nine years of industry experience ($mdn = 7$) across professionals and students. Participants self-reported spending approximately 6.5 hours per week ($mdn = 4$) preparing for technical interviews. Most participants ($n = 108$, 82.4%) completed at least one technical interview, averaging three per participant ($mdn = 4$). Since our focus is tech interview preparation, we include participants who are preparing but have not yet completed an interview.

5 Results

5.1 RQ1: Technical Interview Preparation

To understand how candidates prepare for technical interviews, we examined the resources participants use to prepare and how often training settings reflect authentic tech interview environments.

5.1.1 Resources. We found participants use a wide array of technical interview preparation resources are available to help candidates, representing varied mediums with different features. For example, there are online coding practice platforms (*i.e.,* LeetCode [6]–depicted in Figure 1), study guides (*i.e.,* Blind75 [40]), and physical books (*i.e.,* "*Cracking the Coding Interview*" [78]). YouTube [10], ($n = 108$), LeetCode [6] ($n = 90$), and GeeksforGeeks [2] ($n = 77$) were the most popular resources reported and the most frequently used among participants (see Figure 2). Other resources mentioned include Exponent, Pramp, Hackerrank, Udemy, Coursera, Algomonster, educative.io, AlgoExpert, Stack Overflow, and more. Eight participants reported using no resources. We also found candidates are unwilling to pay for tech interview preparation resources, with few participants reporting being likely to subscribe to premium accounts for LeetCode ($n = 28$), Exponent ($n = 12$), and Grokking the Coding Interview ($n = 10$), or purchase the Cracking the Coding Interview book ($n = 30$).

5.1.2 Authenticity. To investigate the authenticity of tech interview preparation, we evaluated the extent to which participants practice: (1) coding; (2) communication; and (3) with observers.

Coding. Technical interviews are designed to assess the coding abilities and technical knowledge of candidates. Generally, the preparation resources used by participants focus on coding practice for technical interviews. For example, participants who reported using LeetCode spend approximately two hours each day practicing data structures and algorithms-related programming challenges. Moreover, over 85% of participants rank problem-solving ability as *Very Important* ($n = 85$) or *Important* ($n = 27$) when preparing for tech interviews. On average, participants reported specifically practicing LeetCode challenges around two hours per day ($mdn = 1$) and three days per week ($mdn = 2$).

Communication. Communication is essential in software engineering [68]. As such, think-aloud is an integral part of technical interviews [47], sometimes regarded as more important than coding ability [26]. Our results suggest candidates view communication as critical in technical interviews, with 78% of participants reporting oral and verbal clarity in interviews as *Important* ($n = 39$) or *Very Important* ($n = 63$). However, most participants ($n = 39$) reported only practicing communication skills when preparing for technical interviews *Occasionally*, with less than half practicing think-aloud *Sometimes*, *Often*, or *Always* (see Table 1).

Audience. Another key part of technical interview environments is coding and communicating *in front of an observer*. However, we found participants rarely practice with others—as 62% ($n = 81$) responded *No* to a question asking if their technical interview preparation includes studying with others. Candidates who reported preparing with others ($n = 50$) mentioned various approaches, including training with friends, family members, colleagues, and study groups in-person and remotely using platforms such as Discord and Zoom. We further investigated the types of preparation with others. Candidates can practice with experts—for example, many preparation resources offer features for users to receive coaching from experts. Yet, participants rarely prepare with tutors (29%, $n = 38$) and fewer pay for coaching functionalities (31.6%, $n = 12$). We also explored participants' experience with mock interviews, or simulated technical interviews with peers. However, mock interviews are infrequently used ($avg = 4.6$, $mdn = 2$)—with most respondents (82%, $n = 107$) completing five or less and 35% ($n = 46$) never attempting a mock interview (see Figure 3).

Finding 1: Most participants use free, online resources (*i.e.,* YouTube, LeetCode, and GeeksforGeeks) to prepare for technical interviews.
Finding 2: We found most candidates do *not* prepare for technical interviews in authentic settings—prioritizing coding practice but rarely preparing in front of an audience with observers.

5.2 RQ2: Effects of Tech Interview Preparation

To analyze candidates' perceptions of interview preparation, we investigated their self-reported anxiety during interview training and preparedness for interviews.

5.2.1 Anxiety. We found most participants ($n = 77$, 58.8%) reported feeling anxious during tech interview preparation. Our qualitative analysis uncovered numerous reasons participants felt anxious while training for interviews, presented in Table 2). The main causes were candidates having to study too many topics and the uncertainty of what interviewers will ask. For example, P82 noted "*It's mostly the daunting task of trying to remember certain data structures and what you have to recollect and learn again because it is not part of your day to day operations*" and P131 added "*There's just so much interviewers could ask you and I simply do not know all of it*". Another source of anxiety was candidates not having enough time to prepare given their other commitments. For instance, participants noted difficulties regarding "*the amount of time required to study for interview and the busy working day as I worked as a full-time*" (P8) and "*Balancing the amount to time you need to invest to be able*

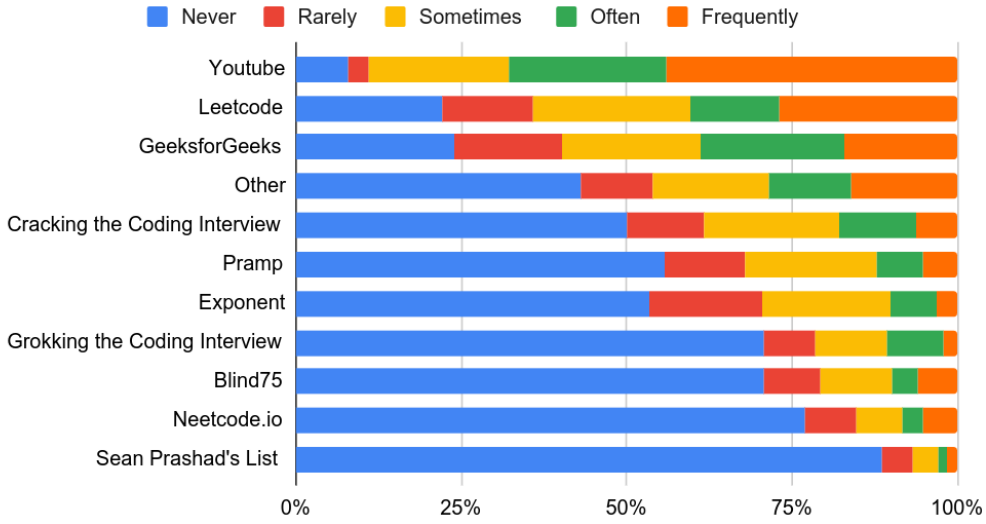

Figure 2: Frequency of reported resources used

Table 1: How often do you practice your communication skills during your technical interview prep session?

Response	Never	Rarely	Occasionally	Sometimes	Often	Always
n (%)	18 (13.7%)	11 (8.4%)	39 (29.8%)	8 (6.1%)	32 (24.4%)	23 (17.6%)

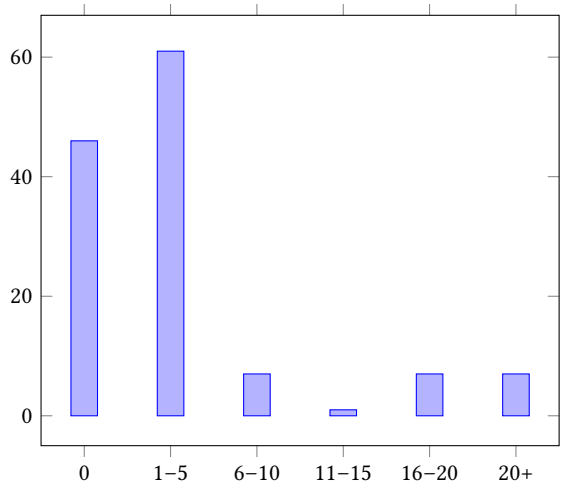

Figure 3: Range of responses for the number of mock interviews participants have completed

Table 2: Causes of Anxiety in Tech Interview Preparation

Response	***Participants (n)***
Too many topics to study	73
Unsure on what will be asked	73
Balancing of prep and other time commitments	69
Poor performance in past interviews	52
College did not prepare me	43
Other	70
None	8

* Total exceeds number of participants because multiple reasons could be provided in a single response.

Table 3: Perceived Anxiety Based on Participant Background and Preparation

Background	Type	Median	*p*-value
Gender	Female	4	$p = 0.2148$
	Male	4	($U = 1246$, $\eta^2 = 0.14$)
Race	non-White	3	$p = 0.3121$
	White	4	($U = 1329.5$, $\eta^2 = 0.47$)
Status	Student	4	$p = 0.3632$
	non-Student	4	($U = 1798$, $\eta^2 = 0.06$)
Interviews	Lower Half	4	$p = 0.1075$
	Upper Half	3	($U = 1846.5$, $\eta^2 = 0.01$)
Preparation	**Type**	**Median**	***p*-value**
Hours	Lower Half	4	$p = 0.3446$
	Upper Half	4	($U = 2027$, $\eta^2 = 0.001$)
LeetCode	Lower Half	4	$p = 0.2327$
	Upper Half	4	($U = 2785.5$, $\eta^2 = 0.08$)
Communication	Frequent	4	$p = 0.3897$
	Infrequent	4	($U = 2051.5$, $\eta^2 = 0.001$)
With Others	Yes	4	$p = 0.2033$
	No	4	($U = 1850$, $\eta^2 = 0.15$)

* denotes statistically significant results (**p-value < 0.05**), η^2 is effect size

Table 4: Perceived Preparedness Based on Participant Background and Preparation

Background	Type	Median	*p*-value
Gender*	Female	3	$p = 0.0091^*$
	Male	3	($U = 989$, $\eta^2 = 0.04$)
Race	non-White	3	$p = 0.3121$
	White	3	($U = 1509$, $\eta^2 = 0.004$)
Status*	Student	4	$p = 0.0418^*$
	non-Student	4	($U = 1743$, $\eta^2 = 0.02$)
Interviews	Lower Half	3	$p = 0.0838$
	Upper Half	3	($U = 1815$, $\eta^2 = 0.02$)
Preparation	**Type**	**Median**	***p*-value**
Hours	Lower Half	3	$p = 0.1587$
	Upper Half	3	($U = 1898$, $\eta^2 = 0.01$)
LeetCode	Lower Half	3	$p = 0.3520^*$
	Upper Half	3	($U = 2029.5$, $\eta^2 = 0.001$)
Communication*	Frequent	3	$p = 0.0222^*$
	Infrequent	3	($U = 1680$, $\eta^2 = 0.03$)
With Others*	Yes	3	$p = 0.0256^*$
	No	3	($U = 1613$, $\eta^2 = 0.35$)

* denotes statistically significant results (**p-value < 0.05**), η^2 is effect size

to get to a good level with school and jobs and personal time is very hard" (P129). The effects of this anxiety also negatively impacted candidates, with participants commenting on preparation, it "*makes me more anxious before the real interview*" (P8), "*I think I'm never prepared or good enough*" (P23), "*I have panic attacks all the time and feel like I don't have enough time to prepare*" (P53), and "*When I was preparing, I always felt guilty when I wasn't working to prepare more. I felt like I could never take a break*". These responses suggest the tech interview preparation techniques reported by participants fail to support candidates actively seeking SE positions, resulting in anxiety, frustration, and burnout from their training efforts.

We further investigated whether participants demographic group and preparation methods impacted perceived anxiety. For background, we compare the majority and minority groups for gender, race and status, while dividing number of interviews completed based on the median. For preparation methods, we observed number of reported study **hours** per week, number of daily **LeetCode** problems, reported **communication** practice frequency—based on frequent (*Always, Often, Sometimes*) and infrequent (*Occasionally, Rarely, Never*) Likert responses, and whether or not participants reported practicing **with others**. These results are presented in Table 3. Using a Mann-Whitney U test (U), we did not find a significant correlation. This is a potential indicator that the anxiety induced by technical interview preparation impacts all candidates, regardless of background or processes used to prepare.

5.2.2 Preparedness. The results for participants' self-reported preparedness are presented in Table 5. Overall, less than 40% ($n = 52$) of participants responded positively to being prepared for interviews, with the majority feeling uncertain about their preparedness.

To further investigate perceived preparedness, we used a Mann-Whitney U to statistically analyze preparedness against participant background and preparation techniques (see Table 4). For demographic background, we observed a statistically significant difference in perceived preparedness based on the gender of participants (male/female, $p = 0.0091$). This suggests female candidates feel significantly less prepared than male-identifying counterparts while preparing for technical interviews. We also observed experience plays a role in candidates' preparedness, as current software engineers reported feeling significantly more prepared for interviews compared to students ($p = 0.0418$).

We also explored whether various practices impact perceived preparedness. We observed more frequently rehearsing communication ($p = 0.0222$) and practicing with others ($p = 0.0256$) were significant indicators in participants' perceived preparedness—suggesting that authentic practice environments, particularly training communication skills and with others, can improve candidates' perceived preparedness for tech interviews. In addition, contrary to prior work and popular trends in technical interview preparation [39, 67, 82], we observed that candidates reporting more study hours per week and completed LeetCode problems per day did *not* perceive to be more prepared for technical interviews.

Finding 3: We observed technical interview preparation invokes anxiety for candidates, primarily due to extensive number of concepts, uncertainty of topics, and lack of time to prepare.
Finding 4: Most candidates feel their preparation methods do *not* prepare them for technical interviews. However, candidates who reported practicing communication more frequently as well as training with others reported significantly higher perceived preparedness—while those who reported completing more daily LeetCode problems and studying more hours per week did not.

5.3 RQ3: SE Education

We asked participants to provide insights on how their university curriculum impacted their tech interview preparation to investigate gaps and highlight competence needs for students pursuing SE roles. When asked whether undergraduate courses helped prepare for interviews, most participants ($n = 64$, 48.9%) responded *No*. We further asked participants to rank how well their courses

Table 5: Participant Responses to "*I feel that I am well prepared for the day of my technical interview*"

Response	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
***n* (%)**	7 (5.3%)	29 (22.1%)	43 (32.8%)	38 (29.0%)	14 (10.7%)

Table 6: Participant Responses to whether University Courses Prepared them for Technical Interview Concepts

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Data Structures	31	31	37	31	14
Algorithms	30	27	41	27	13
Time/Space Complexity	24	30	39	30	19

Table 7: Participant Responses to "*I would have benefited from a structured study plan provided by my university*"

Response	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
***n* (%)**	14 (10.7%)	12 (9.2%)	25 (19.1%)	30 (22.9%)	50 (38.2%)

prepared them for concepts common in technical interviews [78], shown in Table 6. Most participants were negative or neutral on whether courses prepared them for data structures (75.6%), algorithms (74.8%), and time/space complexity of programs (71.1%). The most common courses that were mentioned as being helpful include Data Structures, Algorithms, Computer Architecture, capstone course, and more–with one participant stating "*AP Computer Science in High School. It was much more helpful than all the courses I took in college*" (P45). Further, only one participant mentioned an SE course was useful for technical interview preparation.

We also asked participants if tech interview-related courses would aid preparation efforts—with most participants (n = 98, 74.8%) responding a specific tech interview course would be helpful, while 20 (15.3%) and 13 (9.9%) selected *No* and *Unsure*, respectively. We also asked if a structured study plan provided by institutions could improve skills needed for technical interviews. The majority of participants (n = 50, 38.2%) strongly agreed that a university or department study plan would support their technical interview preparation efforts (see Table 7). These results provide insights into methods to enhance computing education to better support technical interview preparation for candidates.

Finding 5: Most candidate report university-level CS courses did *not* adequately prepare them for technical interview concepts, such as data structures and algorithms.
Finding 6: Most candidates believe structured study guides provided by universities can benefit technical interview preparation efforts.

6 Discussion

Our results suggest candidates prepare for technical interviews using a variety of different resources, yet rarely practice in authentic environments with coding, communication, and observers (RQ1). Consequently, most participants feel their preparation leaves them anxious and unprepared for interviews (RQ2). Participants also reported a lack support from their computing education (RQ3). Based on our results, we provide guidelines for how candidates, employers, tech interview preparation resources, and higher education can overcome difficulties in technical interview preparation.

6.1 For Candidates

Our findings show that candidates frequently practice coding skills and often rehearse communication during technical interview preparation. However, participants rarely prepared with other people. Beyond increasing the amount of time and effort to prepare [39], which our results suggest may not impact preparedness for interviews (see Table 4 in Section 5.2), we posit preparation with others can help candidates enhance learning for interview skills and feel better prepared for hiring assessments.

6.1.1 Practice with Others. Our results show that candidates who prepare with others feel significantly more prepared for their technical interviews than those who practice alone. The public nature of technical interviews has been shown to increase stress and worsen interview performance [23]. Prior studies also show participants exhibit less stress in interview environments without an observer present [21, 22]. However, interviewer presence is common in interviews [93]. Thus, preparing with other people, using approaches such as mock interviews, can enhance preparation for actual interview performances. For example, mock interviews, which were underutilized by candidates in our study, can simulate tech interviews with time pressure, a live observer, and think-aloud communication.

Mock interviews are also a form of *collaborative learning*, or an educational approach where students work together in groups to learn and solve problems [64], which has been shown to enhance outcomes and learning for students in a variety of domains [63] and SE contexts [36, 37]. For instance, collaborative learning through mock interviews can preparation beyond individual practicing through pooled knowledge, explaining concepts, error-correction, reduced memory load, and observational learning [81]. Thus, practicing with others can provide more relevant experience with authentic environments for technical interviews during preparation—increasing students' learning and confidence in skills necessary to succeed in technical interviews.

6.2 For Employers

While improving technical interview preparation will help candidates, it can also benefit companies. Hiring processes are time-consuming and expensive [42]. To that end, we provide implications for companies to help candidates better prepare and motivate improvements to SE hiring practices.

6.2.1 Rethink Technical Evaluations. We found technical interview preparation anxiety-inducing, regardless of candidate experience. Further, increased preparation efforts do not enhance candidates' perceived preparedness for assessments. Thus, modifications to SE hiring processes are needed to focus candidates' preparation, providing better evaluations of candidates' abilities and their suitability for a role at the company. For example, think-aloud allows employers to gain insight into the thought processes of candidates—however, this is frequently combined with evaluating soft skills [21]. We also found candidates infrequently practice communication skills during interview prep, and even more rarely practice in front of an observing audience (see Section 5.1.2). For instance, P25 noted the unfairness of this evaluation approach, stating employers "*just expected [them] to talk it out like you work there...How is that apple to apples?*". P100 also noted they lack "*confidence in putting thoughts to words*". Thus, separating think-aloud and soft skills assessments can better assess candidates' fit based on their experiences, character, personality traits, aptitude, and communication ability.

Another concern is the irrelevance of technical interviews to real-world SE work [20]. For instance, in our survey P4 noted that tech interview prep is "*a different skill set to real work and you need to be able to memorize solutions which you won't encounter in the real world*". As such, candidate preparation often does not prioritize learning skills necessary to succeed as a software engineer [68], but focuses on learning concepts needed to pass the interview. P45 highlights this frustration, stating, "*sadly, many LeetCode questions suffer from you-either-know-it-or-you-dont problem...If you don't know the algorithm, you won't be able to come up with an optimal solution. It has taken geniuses years to come with up with a solution, so how could you be expected to come up with it in 30 minutes?*". This can be problematic for companies, as "LeetCoders", or candidates who memorize solutions to whiteboard-style technical interview problems without necessarily understanding the implementation, can deceive employers due to good tech interview performances [96]. However, these candidates may not perform well in a real-world development environment. In addition, these false positive hires can cause employers to miss out on qualified candidates who may not have performed as well on the coding challenge during the interview. Thus, providing more relevant assessments can improve candidates' preparation and evaluation—thereby improving the quality of software and the overall tech workforce.

6.2.2 Provide Feedback on Interview Performance. Another challenge in tech interview preparation is the lack of feedback [22]. Candidates typically do not receive feedback on interview performances, making it difficult to improve for future interviews. Companies avoid giving feedback for various reasons, such as policies and legal implications [56] and a lack of infrastructure to provide feedback [70]. However, feedback is critical for enhancing learning [88]. For instance, research shows the act of processing feedback leads to better performance [71]. In addition, a lack of feedback has been shown to inhibit learning and cause students to become discontented [28]. In our survey, we saw a lack of feedback from technical interviews led to anxiety and frustration for participants, such as one who noted "*Saving time for interview prep then getting rejected is exhausting and disappointing*" (P115) and another stating "*I cannot figure out what the interviewer is looking for*" (P23).

Other forms of feedback—such as comments on code reviews—has shown to increase learning in SE settings [18]. Automatically generated feedback has also shown promise in increasing student learning and programming behaviors [45]. Recently, tools have employed large language models (LLMs) to automatically generate feedback on mock interview performances [29]. However, research shows that candidates distrust LLMs in hiring contexts [98], in addition to technical interview settings [97]. Further, specific feedback from interviewers can help candidates gain insights into the expectations of employers [47], providing insights on ways to improve their interview performance and lead to potential job offers the same company or a different organization.

6.3 Technical Interview Preparation Resources

Our survey results show candidates use a wide variety of resources for technical interview preparation (see Figure 2). As the goal of these systems should be to help candidates enhance their skills to obtain job offers, we provide guidelines to improve existing technical interview preparation resources and motivate the design of future systems to enhance candidates' preparation processes.

6.3.1 Remove financial barriers. Most technical interview preparation resources primarily focus on practicing coding skills, providing ample programming challenges based on various data structures and algorithms for candidates to solve. Many training resources also provide social and collaborative preparation methods. However, these features are often hidden behind financial barriers that require paid accounts. For instance, LeetCode Premium costs $35 per month [7], expert feedback on IGotAnOffer costs $149 per hour [4], and premium interviews for interviewing.io start at $225 per interview [5]. Our results show most candidates are unwilling to pay and often rely on free resources, such as YouTube, LeetCode (free version), and GeeksforGeeks. Financial costs also present a barrier to learning, particularly for individuals from disadvantaged groups [83]. Thus, reducing financial barriers to more advanced features can improve candidates' preparedness and self-efficacy for technical interview evaluations.

6.3.2 Provide opportunities for authentic training and spectatorship. Our results show that most candidates do not train in authentic technical interview settings—prioritizing technical skills practice over communication and social aspects of tech interviews. However, we observed candidates who practice communication more frequently and prepare with others felt significantly more prepared for technical interviews. Thus, future systems should seek to integrate functionality for authentic interview settings, such as think-aloud practice and training with others—which we found significantly improves candidates' perceived preparedness. For example, in our study we observed candidates who practice with others reported using virtual platforms, such as Discord, which has been touted as an alternative online learning medium [17]. In addition, Pramp provides functionality that matches users together to complete mock interviews between peers [8], promoting authentic practice and collaborative learning. Recent work has also explored using artificial intelligence and LLMs to simulate general interviews [13, 29, 34, 69] and technical interviews [15, 46, 97] in support of candidates' preparation for job assessments.

Another way technical interview preparation platforms can enhance candidates' preparation efforts is by fostering *spectatorship*, or the act of watching others complete a task. We were surprised to find YouTube is the most frequently used resource for technical interview preparation reported by participants, contrary to prior work suggesting LeetCode is the most popular [39]. However, studies show YouTube is commonly used for learning programming concepts by diverse learners [35, 59]. An advantage for YouTube is that it supports spectatorship—such as watching YouTube videos of users solve programming problems, which can support learning. For instance, prior work investigates spectatorship in creative writing, and found that users watching other writers helped them understand and improve their own writing processes [30]. Spectatorship of gameplay videos on YouTube enhances information sharing and learning—helping watchers adopt new techniques and strategies for their own video games [49]. Live streaming, where individuals broadcast their work to live viewers, is also an emerging practice for scaling CS education [32] and understanding software development practices [14]. Incorporating these functionalities into tech interview training resources can enhance candidates' learning and preparation for job assessments.

6.4 SE Education

Finally, we provide guidelines to improve SE education to enhance tech interview preparation for candidates. We specifically focus on university-level computing education within academic institutions. One of the core responsibilities of universities is to shape students through the education of their chosen discipline. While several participants in our survey were not from CS backgrounds, we focus on enhancing CS and SE education as they are tasked with preparing students for computing careers and represent the majority of candidates completing technical interviews. However, due to the dynamic nature of computing, SE education requires constant adjustment to meet the evolving needs of the tech industry [48].

6.4.1 Integrate authentic interview training in classrooms. SE courses are intended to provide students with the knowledge and skills to succeed as software engineers [91]. However, only one survey participant mentioned an SE course was useful for their technical interview preparation—with most noting Data Structures and Algorithms-related courses as more helpful. Further, most participants found their undergraduate curriculum in general not useful for preparing for technical interviews. Thus, computing curricula can be enhanced to support candidates pursuing SE jobs. Prior work explores adding technical interview specific courses or interview practice into existing CS courses [51, 60]. While these lack the high stakes nature and anxiety-inducing nature of real job interviews [61], they can provide realistic environments and social settings to train for interviews.

6.4.2 Incorporate SE-related Activities to Support Authentic Interview Concepts. In addition, SE courses in particular can provide opportunities for authentic interview practice—coding and communication in front of an audience—beyond typical data structures and algorithms-focused classes. Software development is a collaborative activity, relying on teams of developers to implement and maintain software [99]. SE-related classes often provide opportunities for collaborative learning through group projects [38], which has been shown to enhance learning experiences [77] and promote effective communication among students [76]. Prior work also suggests incorporating practices from Agile—a development processes commonly used in the tech industry, such as scrum meetings [102] and retrospectives [62], can enhance communication and soft skills for students in SE-related classes [43].

Moreover, a close environment to authentic interview settings is pair programming, where two software engineers work together on the same machine to write code [25]. While pair programming lacks the evaluative nature of tech interviews, Williams *et al.* suggest this practice is effective in educational contexts for enhancing student learning and improving communication [100]. Further, prior work suggests incorporating pair programming in technical interviews—between candidates and interviewers—can reduce anxiety and provide insights into candidates' technical abilities and communication skills in a more relevant setting [21].

6.4.3 Avoid Extracurricular Technical Interview Preparation Only Focused on Coding. Our results suggest the lack of time to prepare for technical interviews leads to anxiety among candidates (see Table 2). This aligns with prior work, where candidates and developers decry the amount of time and effort needed to prepare for tech interviews [20]. We also found that increasing study efforts does not enhance candidates' preparedness (see Table 4). Based on our findings, we *discourage* implementing co- and extracurricular activities—learning activities outside of the classroom—to support technical interview preparation. Extracurricular activities have been shown to enhance academic performance and provide additional learning opportunities for students [41]. This has also been proposed as a method to address gaps in SE education and academia in prior work [55]. Yet, while we observed increased study time improves perceived preparedness, we also found a lack of time to complete tasks was a major cause of anxiety in tech interview preparation. For example, P35 noted the difficulties of "*balancing interview prep time with other commitments like coursework*" led to anxiety in their preparation. Further, prior work suggests students—particularly those from minority backgrounds [73]—may have difficulty balancing interview preparation with coursework, extracurricular activities, and other aspects of their lives. For instance, P70 noted they "*can't focus [on] during work days. Can only do [preparation] on weekends*". In addition, we observed female participants felt significantly less prepared for interviews—despite spending more hours per week (7.37) on average preparing than males (6.71). Thus, additional activities may increase anxiety or stress for candidates seeking SE positions.

Alternatively, extracurricular activities that do support technical interview preparation should include authentic preparation. For instance, clubs and study groups focused on solving LeetCode problems are popular at a wide variety of institutions.[4] However, our results found that candidates who report practicing more LeetCode problems per day did not feel more confident in their preparation for interviews—while candidates with more communication practice and training with others had higher perceived preparedness (see Table 4). Thus, extracurricular clubs and organizations

[4] https://leetcode.com/student/

focused on technical interview preparation should incorporate authentic interview environments, by integrating activities such as mock interviews and promoting spectatorship. Moreover, other computing-related activities can support tech interview training and job attainment. For instance, prior work shows underrepresented students often rely on minority-focused groups, such as the National Society of Black Engineers (NSBE), for networking and interview preparation [75]. These groups also help minority candidates combat microaggressions and enhance their computing identity and sense of belonging [75, 87].

Other technical-focused activities have also shown to support enhancing soft skills. For instance, the goal of competitive programming is to help students learn and apply algorithms—enhancing their computational thinking, problem-solving, and programming skills [101]—and Moreno *et al.* suggest competitive programming teams can help students practice synchronous communication and team problem-solving abilities [79]. Hackathons, social coding events where participants develop new software, have also been shown to promote community-based learning [65], enhancing technical abilities and soft skills such as teamwork and communication [89]. Enhancing opportunities for these activities can help students gain practice with communication and being in front of an audience—increasing their performance in tech interview settings.

6.4.4 Provide a Department-Specific Study Guide. We found participants believe a study guide from their university would be helpful to support tech interview preparation (see Table 7. Thus, CS departments can also explore producing a study guide or curriculum to support students pursuing SE positions specifically. Technical interviewing guidelines exist broadly (*i.e.,* Cracking the Coding Interview [78] and the Tech Interview Handbook [9]) and for specific companies (*i.e.,* Google [86] and Amazon [84]). However, university-specific guidelines can help align preparation efforts with specific courses to support students' career goals. For instance, Duke University published a general technical interviewing guide to provide an overview of what students should expect [85]. This could also benefit student job placement, a key metric for schools [12]. Further, instructors play a role in students' industry and interview preparedness [57]. Lunn *et al.* show CS instructors desired institutions to provide encouragement and information to raise awareness of technical interview practices—in addition to providing instructor training and resources [74]. Alternatively, learning what concepts are not covered in computing curricula can highlight gaps and motivate learning outside of the classroom [68].

Study plans should also be embedded within curricula—as departments vary widely in terms of available classes, topics, etc. [11]. For instance, data structures such as hash tables and stacks—commonly used in technical interviews [78]—are infrequently taught and assessed in CS2-level data structures courses [92]. A department-specific study guide could also provide insights on courses to take based on students' career goals. For instance, certain departments may have specialized classes focused topics such as web development [33], cloud computing [31], and other advanced SE-related concepts. Customized study plans can help guide students through computing curricula, providing knowledge to help them succeed in tech interviews and attain computing careers.

7 Limitations

There are several threats to the validity of this work. Technical interview processes and preparation vary widely on different factors—such as company, role, etc. In this work, we only focus on whiteboard-style tech interviews—commonly used to evaluate candidates for a wide variety of tech roles [93]. Future work is needed to explore preparation for other techniques—such as take-home assessments [95]. Additionally, while we recruited a diverse sample of candidates actively preparing for tech interviews in pursuit of SE roles, our results may not generalize to all candidates on the tech job market. For instance, we primarily targeted users of two platforms—Exponent [1] and Pramp [8]. However, our results found these participants used a wide range of resources beyond these tools. Our survey excludes additional factors that may impact preparation, *i.e.,* GPA, supportive home environments, etc. However, our goal was to investigate the practices, authenticity, and educational impacts on technical interview preparation. Our survey also relies on self-reported data, such as study hours per week, number of LeetCode problems completed per day, perceived anxiety, and perceived preparedness, which may be inaccurate and biased.

8 Future Work

Future work can explore specific technical interview preparation resources and practices to investigate how their usage and impact on preparedness. For example, analyzing the content of tech interview-related YouTube videos. Observational approaches can provide further details about how candidates prepare for technical interviews and the resources they use. Additionally, longitudinal studies can provide additional insights on the effects of preparation, such as the types of roles candidates interview for and whether or not they receive a job offer. Based on our findings, we plan to implement novel training resources (Section 6.3) and explore SE curricula changes and development (Section 6.4) to support authentic interview preparation for candidates joining the tech workforce.

9 Conclusion

Technical interviews are a socially and cognitively demanding activity, leading to considerable preparation time and effort for aspiring software engineers. To understand how candidates prepare for technical interviews, we distributed an online survey to collect data from 131 individuals actively preparing for technical interviews in pursuit of SE-related roles. Our results show that participants use a variety of resources such as YouTube and LeetCode, but rarely practice in authentic interview settings. We also found SE education is inadequate for supporting technical interview preparation, and most participants perceive their preparation efforts cause anxiety and fail to prepare them for actual interviews. Based on our findings, we provide implications for enhancing technical interview preparation—providing guidelines for candidates, employers, interview prep materials, and SE education to improve candidates' preparedness for the complexity of tech hiring processes.

10 Acknowledgments

We would like to thank Stephen Cognetta and Lindsey Parker at Exponent/Pramp for their insights and help distributing our survey. This work was supported by a Google Award for Inclusion Research.

References

[1] [n. d.]. Exponent. https://www.tryexponent.com/.
[2] [n. d.]. GeeksforGeeks. https://www.geeksforgeeks.org/.
[3] [n. d.]. Glassdoor. https://www.glassdoor.com.
[4] [n. d.]. IGotAnOffer. https://igotanoffer.com/.
[5] [n. d.]. interviewing.io FAQ. https://interviewing.io/faq.
[6] [n. d.]. LeetCode. https://leetcode.com.
[7] [n. d.]. LeetCode Premium. https://leetcode.com/subscribe/?ref=&source=ot.
[8] [n. d.]. Pramp. https://www.pramp.com/.
[9] [n. d.]. Tech Interview Handbook. https://www.techinterviewhandbook.org/.
[10] [n. d.]. YouTube. https://www.youtube.com/.
[11] 2023. Computer Science Programs: How to Differentiate. *College Confidential* (2023). https://talk.collegeconfidential.com/t/computer-science-programs-how-to-differentiate/3641245.
[12] 2024. Colleges with Strong Computer Science Job Placement. *College Vine* (2024). https://www.collegevine.com/faq/159543/colleges-with-strong-computer-science-job-placement.
[13] Lalit Agrawal, Pritam Lanjewar, Shirisha Deshpande, Parag Jawarkar, Vandana Gaur, and Aditya Dive. 2024. The Impact of AI on Communication Skills Training Opportunities and Challenges. *Nanotechnology Perceptions* (2024), 1167–1173.
[14] Abdulaziz Alaboudi and Thomas D LaToza. 2023. What constitutes debugging? An exploratory study of debugging episodes. *Empirical Software Engineering* 28, 5 (2023), 117.
[15] Aptico. 2024. Aptico: AI-Powered Interview Preparation. https://www.aptico.xyz/ Accessed: 2025-01-09.
[16] Andrea Arcuri and Lionel Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd international conference on software engineering*. 1–10.
[17] Muhammad Arifianto and Iqbal Izzudin. 2021. Students' acceptance of discord as an alternative online learning media. *International Journal of Emerging Technologies in Learning (iJET)* 16, 20 (2021), 179–195.
[18] Alberto Bacchelli and Christian Bird. 2013. Expectations, outcomes, and challenges of modern code review. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 712–721.
[19] 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. https://doi.org/10.1145/3183399.3183415
[20] Mahnaz Behroozi, Chris Parnin, and Titus Barik. 2019. Hiring is broken: What do developers say about technical interviews?. In *Visual Languages & Human-Centric Computing (VL/HCC)*. 1–9. https://doi.org/10.1109/VLHCC.2019.8818836
[21] 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)*. Association for Computing Machinery, New York, NY, USA, 294–305. https://doi.org/10.1145/3540250.3549168
[22] 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
[23] 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.
[24] Brian Alexander Bell. 2023. *Understanding the Preparation Phase of Technical Interviews*. Master's thesis. Virginia Tech.
[25] Tanja Bipp, Andreas Lepper, and Doris Schmedding. 2008. Pair programming in software development teams–An empirical study of its benefits. *Information and Software Technology* 50, 3 (2008), 231–240.
[26] Laurence Bradford. 2020. Technical Interviewing 101: Ultimate Guide to Acing Your Tech Interview in 2021. https://learntocodewith.me/posts/technical-interview/.
[27] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
[28] James Brown. 2007. Feedback: the student perspective. *Research in Post-Compulsory Education* 12, 1 (2007), 33–51.
[29] Career.io. 2024. Career.io: End-To-End Career Services. https://career.io/ Accessed: 2025-01-09.
[30] Dashiel Carrera and Sang Won Lee. 2022. Watch Me Write: Exploring the Effects of Revealing Creative Writing Process through Writing Replay. In *Creativity and Cognition* (Venice, Italy) *(C&C '22)*. Association for Computing Machinery, New York, NY, USA, 146–160. https://doi.org/10.1145/3527927.3532806
[31] Ling Chen, Yang Liu, Marcus Gallagher, Bernard Pailthorpe, Shazia Sadiq, Heng Tao Shen, and Xue Li. 2012. Introducing cloud computing topics in curricula. *Journal of Information Systems Education* 23, 3 (2012), 315.
[32] Yan Chen, Walter S. Lasecki, and Tao Dong. 2021. Towards Supporting Programming Education at Scale via Live Streaming. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3, Article 259 (Jan. 2021), 19 pages. https://doi.org/10.1145/3434168
[33] Chien Chou and Chin-Chung Tsai. 2002. Developing web-based curricula: Issues and challenges. *Journal of curriculum studies* 34, 6 (2002), 623–636.
[34] Yi-Chi Chou, Felicia R Wongso, Chun-Yen Chao, and Han-Yen Yu. 2022. An AI mock-interview platform for interview performance analysis. In *2022 10th International Conference on Information and Education Technology (ICIET)*. IEEE, 37–41.
[35] Yousra Chtouki, Hamid Harroud, Mohammed Khalidi, and Samir Bennani. 2012. The impact of YouTube videos on the student's learning. In *2012 international conference on information technology based higher education and training (ITHET)*. IEEE, 1–4.
[36] Peter J Clarke, Debra Davis, Tariq M King, Jairo Pava, and Edward L Jones. 2014. Integrating testing into software engineering courses supported by a collaborative learning environment. *ACM Transactions on Computing Education (TOCE)* 14, 3 (2014), 1–33.
[37] Mauro Coccoli, Lidia Stanganelli, and Paolo Maresca. 2011. Computer supported collaborative learning in software engineering. In *2011 IEEE global engineering education conference (EDUCON)*. IEEE, 990–995.
[38] David Coppit and Jennifer M Haddox-Schatz. 2005. Large team projects in software engineering courses. *ACM SIGCSE Bulletin* 37, 1 (2005), 137–141.
[39] 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)*. IEEE, 1–10.
[40] Krishna Dey. [n. d.]. Blind 75 LeetCode Questions. LeetCode. https://leetcode.com/discuss/general-discussion/460599/blind-75-leetcode-questions.
[41] Ariane Díaz-Iso, Almudena Eizaguirre, and Ana García-Olalla. 2019. Extracurricular activities in higher education and the promotion of reflective learning for sustainability. *Sustainability* 11, 17 (2019), 4521.
[42] Nathan Doctor. 2016. The Hidden Cost of Hiring Software Engineers — $22,750/hire. Qualified Blog. https://www.qualified.io/blog/posts/the-hidden-cost-of-hiring-software-engineers.
[43] Manoj Joseph D'Souza and Paul Rodrigues. 2015. Extreme pedagogy: An agile teaching-learning methodology for engineering education. *Indian Journal of Science and Technology* 8, 9 (2015), 828.
[44] Anna M Dybas. 2020. Leetcode Two Sum Walkthrough. *Medium* (2020). https://medium.com/@a.m.dybas/leetcode-two-sum-walkthrough-db7cc389cd48.
[45] Stephen H Edwards. 2003. Rethinking computer science education from a test-first perspective. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. 148–155.
[46] Final Round AI. 2024. Final Round AI: Interview Copilot. https://www.finalroundai.com/ Accessed: 2025-01-09.
[47] Denae Ford, Titus Barik, Leslie Rand-Pickett, and Chris Parnin. 2017. The tech-talk balance: What technical interviewers expect from technical candidates. In *International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 43–48.
[48] Carlo Ghezzi and Dino Mandrioli. 2006. The Challenges of Software Engineering Education. 115–127. https://doi.org/10.1007/11949374_8
[49] Urša Golob, Medeja Kraševec, and Tanja Oblak Črnič. 2021. Video gaming spectatorship: What drives gameplay watching on YouTube? *Media Studies* 12, 23 (2021), 40–56.
[50] Daniel González-Morales, Luz Marina Moreno De Antonio, and José Luis Roda García. 2011. Teaching "Soft" skills in software engineering. In *2011 ieee global engineering education conference (educon)*. IEEE, 630–637.
[51] Jean Griffin, Legand Burge, Sally Goldman, Diego Aguierre, Juan Alonso Cruz, April Alvarez, Albert Cervantes, Shameeka Emanuel, Ann Gates, Daniel Gillick, et al. 2022. Innovative Courses that Broaden Awareness of CS Careers and Prepare Students for Technical Interviews. *Journal of Computing Sciences in Colleges* 38, 5 (2022), 54–64.
[52] 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*. 64–69.
[53] Jocelyn Harper. 2022. Interview insight: how to get the job. In *A Software Engineer's Guide to Seniority: A Guide to Technical Leadership*. Springer, 19–28.
[54] Ewen Harrison and Riinu Pius. 2021. 8.6 Should I convert a continuous variable to a categorical variable? In *R for Health Data Science*. Chapman & Hall/CRC.
[55] Sara Hooshangi, Ryan Buxton, and Margaret Ellis. 2022. Integration of Practical Computing Skills and Co-Curricular Activities in the Curriculum. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (Dublin, Ireland) *(ITiCSE '22)*. Association for Computing Machinery, New York, NY, USA, 61–67. https://doi.org/10.1145/3502718.3524802
[56] Swaminathan Iyer. 2024. 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.

[57] William Gregory Johnson, Raj Sunderraman, and Anu G Bourgeois. 2020. Teaching strategies in software engineering towards industry interview preparedness. In *Proceedings of the 9th Computer Science Education Research Conference*. 1–11.

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

[59] Arbana Kadriu, Lejla Abazi-Bexheti, Hyrije Abazi-Alili, and Veland Ramadani. 2020. Investigating trends in learning programming using YouTube tutorials. *International Journal of Learning and Change* 12, 2 (2020), 190–208.

[60] 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.

[61] 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*. 882–888.

[62] Birgit R Krogstie and Monica Divitini. 2009. Shared timeline and individual experience: Supporting retrospective reflection in student software engineering teams. In *2009 22nd Conference on Software Engineering Education and Training*. IEEE, 85–92.

[63] Marjan Laal and Seyed Mohammad Ghodsi. 2012. Benefits of collaborative learning. *Procedia-social and behavioral sciences* 31 (2012), 486–490.

[64] Marjan Laal and Mozhgan Laal. 2012. Collaborative learning: what is it? *Procedia-Social and Behavioral Sciences* 31 (2012), 491–495.

[65] Miguel Lara and Kate Lockwood. 2016. Hackathons as community-based learning: A case study. *TechTrends* 60, 5 (2016), 486–495.

[66] 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.

[67] Aline Lerner. 2024. We analyzed 100K technical interviews to see where the best performers work. Here are the results. *interviewing.io* (2024). https://interviewing.io/blog/we-analyzed-100k-technical-interviews-to-see-where-the-best-performers-work-here-are-the-results.

[68] 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. IEEE, 700–710.

[69] Sok Ying Liaw, Jian Zhi Tan, Siriwan Lim, Wentao Zhou, John Yap, Rabindra Ratan, Sim Leng Ooi, Shu Jing Wong, Betsy Seah, and Wei Ling Chua. 2023. Artificial intelligence in virtual reality simulation for interprofessional communication training: mixed method study. *Nurse education today* 122 (2023), 105718.

[70] Yi Lu. 2024. *Helping job seekers prepare for technical interviews by enabling context-rich interview feedback*. Ph. D. Dissertation. Virginia Tech.

[71] Caroline Di Bernardi Luft. 2014. Learning from feedback: The neural mechanisms of feedback processing facilitating better performance. *Behavioural brain research* 261 (2014), 356–368.

[72] Stephanie Lunn, Monique Ross, Zahra Hazari, Mark Allen Weiss, Michael Georgiopoulos, and Kenneth Christensen. 2021. The Impact of Technical Interviews, and other Professional and Cultural Experiences on Students' Computing Identity. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. 415–421.

[73] Stephanie J Lunn. 2021. Uneven playing field: Examining preparation for technical interviews in computing and the role of cultural experiences. In *2021 ASEE Virtual Annual Conference Content Access*.

[74] 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*.

[75] 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 Transactions on Computing Education* 24, 1 (2024), 1–29.

[76] 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.

[77] Maira Marques, Sergio F Ochoa, Maria Cecilia Bastarrica, and Francisco J Gutierrez. 2017. Enhancing the student learning experience in software engineering project courses. *IEEE Transactions on Education* 61, 1 (2017), 63–73.

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

[79] Julian Moreno and Andres F Pineda. 2018. Competitive programming and gamification as strategy to engage students in computer science courses. *Revista Espacios* 39, 35 (2018).

[80] Adam Basigie Mtaho and Leonard James Mselle. 2024. Difficulties in learning the data structures course: Literature review. *The Journal of Informatics* 4, 1 (2024), 26–55.

[81] Timothy J Nokes-Malach, J Elizabeth Richey, and Soniya Gadgil. 2015. When is it better to learn together? Insights from research on collaborative learning. *Educational Psychology Review* 27 (2015), 645–656.

[82] Tom Parry. 2023. Coding Interview Prep for FAANG (7 steps to an offer). *IGotAnOffer* (2023). https://igotanoffer.com/blogs/tech/coding-interview-prep.

[83] Jodie Pennacchia, Emily Jones, and F Aldridge. 2018. Barriers to learning for disadvantaged groups. *Report of qualitative findings. Government Social Research. Learning and Work Institute, Department for Education, UK* (2018).

[84] Amazon. [n. d.]. Interiewing at Amazon. *Amazon Jobs* ([n. d.]). https://www.amazon.jobs/content/en/how-we-hire/interviewing-at-amazon.

[85] Duke Career Hub. [n. d.]. Technical Interviewing Guide. *Duke Career Center* ([n. d.]). https://careerhub.students.duke.edu/resources/technical-interviewing-guide/.

[86] Google. [n. d.]. Interiew Prep. *Tech Dev Guide* ([n. d.]). https://techdevguide.withgoogle.com/paths/interview/.

[87] Blanca E Rincón and Sarah Rodriguez. 2021. Latinx students charting their own STEM pathways: How community cultural wealth informs their STEM identities. *Journal of Hispanic Higher Education* 20, 2 (2021), 149–163.

[88] David Roberts. 1996. Feedback on assignments. *Distance Education* 17, 1 (1996), 95–116.

[89] Cleo Schulten and Irene-Angelica Chounta. 2024. How do we learn in and from Hackathons? A systematic literature review. *Education and Information Technologies* (2024), 1–32.

[90] Allison Scott, Freada Kapor Klein, Frieda McAlear, Alexis Martin, and Sonia Koshy. [n. d.]. The Leaky Tech Pipeline: A Comprehensive Framework for Understanding and Addressing the Lack of Diversity across the Tech Ecosystem. *Kapor Center for Social Impact* ([n. d.]). https://leakytechpipeline.com/wp-content/themes/kapor/pdf/KC18001_report_v6.pdf.

[91] Mary Shaw. 2000. Software engineering education: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering*. 371–380.

[92] Beth Simon, Mike Clancy, Robert McCartney, Briana Morrison, Brad Richards, and Kate Sanders. 2010. Making sense of data structures exams. In *Proceedings of the Sixth international workshop on Computing education research*. 97–106.

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

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

[95] Glassdoor Team. 2018. The Surprising Ways Companies Assess Job Applicants. https://www.glassdoor.com/blog/ways-companies-assess-job-applicants/.

[96] Jonathan Tsang. 2021. Why interviewing in tech is broken (and the direction it is going). Evermore. https://jonathantsang.github.io/co-op3/.

[97] Swanand Vaishampayan and Chris Brown. 2025. Will You Trust Me More Than ChatGPT? Evaluating LLM-Generated Code Feedback for Mock Technical Interviews. In *International Conference on Cooperative and Human Aspects of Software Engineering*.

[98] Swanand Vaishampayan, Sahar Farzanehpour, and Chris Brown. 2023. 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)*. IEEE, 103–108.

[99] Jim Whitehead. 2007. Collaboration in software engineering: A roadmap. In *Future of Software Engineering (FOSE'07)*. IEEE, 214–225.

[100] Laurie Williams and Richard L Upchurch. 2001. In support of student pair-programming. *ACM Sigcse Bulletin* 33, 1 (2001), 327–331.

[101] Kevin KF Yuen, Dennis YW Liu, and Hong Va Leong. 2023. Competitive programming in computational thinking and problem solving education. *Computer Applications in Engineering Education* 31, 4 (2023), 850–866.

[102] Sergio Donizetti Zorzo, Leandro De Ponte, and Daniel Lucredio. 2013. Using scrum to teach software engineering: A case study. In *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE, 455–461.