Gaps in Software Engineering Education

Sean M. Gruber

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Dwayne C. Brown, Jr., Chair
Clifford A. Shaffer
Margaret O. Ellis

May 1, 2023
Blacksburg, Virginia

Keywords: Software Engineering, Education, Gaps in Software Engineering Curriculum, Industry, Skills

Gaps in Software Engineering Education

Sean M. Gruber

(ABSTRACT)

Becoming a software engineer can be a stressful process. Software engineers are required to have a broad skill set in order to first obtain a job and then thrive in that position. Job applications will list skills that may be required or recommended but many potential applicants, especially new college graduates, may not have experience with all of the skills that are listed in a position description. The field of software engineering is constantly changing and evolving. New skills are constantly needed in a software engineering position. Education cannot keep up with the constantly changing software engineering workplace. Designing courses takes lots of time and effort. Changing courses to meet the newer and more volatile industry standards could potentially harm existing education by causing a decrease in the quality of more foundation skills. For example, a more foundational skill like good testing practices could become muddled in different languages or frameworks due to a newer language potentially not being fully understood or by the intricacies of the language. This study aims to identify the current gaps that exist between software engineering education and industry. In order to address these gaps this study proposes a platform to provide students with resources related to identified gaps. Additionally, the platform will show the students the identified gaps to see if students are interested in exploring skills related to the identified gaps. The identified gaps are derived from a survey of professional software engineers and subsequent interviews. The results of the study show that students are not only interested in learning how people in industry rank their skills, but that students are overall interested in exploring more resources whether they are directly correlated with a gap or not.

Gaps in Software Engineering Education

Sean M. Gruber

(GENERAL AUDIENCE ABSTRACT)

Becoming a software engineer can be a stressful process. Software engineers are required to have a broad skill set in order to first obtain a job and then thrive in that position. Job applications will list skills that may be required or recommended but many potential applicants, especially new college graduates, may not have experience with all of the skills that are listed in a position description. For example, many applications will list that a specific programming language as a required skill or they may list a specific language framework that is necessary for the job. These skills may not line up with the languages or frameworks that students learn in school. The field of software engineering is constantly changing and evolving. Additionally, the field is so broad that the requirements for different positions can have great variations at different companies. New skills are constantly needed in a software engineering position. Education cannot keep up with the constantly changing software engineering workplace. Designing courses takes lots of time and effort. Changing courses to meet the newer and more volatile industry standards could potentially harm existing education by causing a decrease in the quality of more foundation skills. For example, a more foundational skill like good testing practices could become muddled in different languages or frameworks due to a newer language potentially not being fully understood or by the intricacies of the language. This study aims to identify the current gaps that exist between software engineering education and industry. In order to address these gaps this study proposes a platform to provide students with resources related to identified gaps. Additionally, the platform will show the students the identified gaps to see if students are

interested in exploring skills related to the identified gaps. The identified gaps are derived from a survey of professional software engineers and subsequent interviews. The results of the study show that students are not only interested in learning how people in industry rank their skills, but that students are overall interested in exploring more resources whether they are directly correlated with a gap or not.

Dedication

This work is dedicated to the many who have provided unwavering support to me throughout my academic career. To my fiancée who is constantly providing me with support and motivation through her own excellence. Lastly, to my parents who throughout my entire life have provided me unwavering support and love. This work is not just a culmination of my own effort but the culmination of the effort from all who have supported me.

Acknowledgments

I would like to acknowledge Dr. Chris Brown, Dr. Cliff Shaffer, and Professor Margaret Ellis for serving on my defense committee. Further acknowledgements to Dr. Chris Brown who as my main advisor served as a constant resource for my work and provided advice at every step of my project. I would also like to acknowledge Grace Govan who helped me with various aspects of this project for two semesters. Lastly I would like to acknowledge the entire research group "Code World No Blanket" of which Dr. Chris Brown is the head of. This group provided me a lot of important feedback for several steps of this project.

Contents

List of Figures

List of Tables

List of Abbreviations

SE Software Engineering

Software engineering is a field of computer science where engineering principles are applied to develop and maintain computer software.

Chapter 1

Introduction

Software engineering is a field that is constantly evolving. New software engineering tools are developed every year and different coding languages rise and fall in popularity as time goes on. For example, in the past few months we have seen the rise of Artificial Intelligence tools such as ChatGPT. This is a potentially industry shaking tool as Artificial Intelligence could change the ways the software engineering environment functions. For example, ChatGPT could potentially be used to aid developers in debugging [27]. This work is not focused on the effects of ChatGPT, but ChatGPT is a good example of how quickly the software engineering industry can change. In comparison, software engineering curriculum is much slower to change. This leads to a big disadvantage in that students are not taught the up-to-date practices that are in place in industry, thus making them less prepared for when they start or apply for a full time job. However, this is not without its benefits as well. If software engineering education were to constantly change based on what tools industry is most favoring that year, then the curriculum will most likely become far too imbalanced. Too much focus on ever-changing temporary tools risks taking time away from longer-term skills such as good-design. Good testing practices could become muddled in different languages or frameworks due to a newer language potentially not being fully understood by the course designer or by the intricacies of the language.

This does not just impact schools as companies also have devote a large amount of their

resources to training their employees. Companies spend a lot of resources on training employees as noted by Eric Brechner [2] who says that at one point Microsoft halted all Windows production in order to train their employees on secure coding.

Software engineering education could potentially benefit from attempting to include the newer tools that industry uses. However, since the industry can be so volatile in what are the most popular practices, the inclusion of newer resources needs to be done carefully so as to not lower the existing effectiveness of education. Additionally, there can be other challenges that new languages face that could make them more difficult to integrate into coursework. For example, Chakraborty et al.[5] conduct a study into the growth and challenges of new languages like Swift, Go, Rust, etc., they mention that some of the challenges for these new languages is lack of online resources from previous users. However, as time goes on these newer languages could grow in popularity and become more mainstream within industry. This has been seen throughout the history of software engineering as many of the languages that were used in the sixties are not widely used today and even the popularity of existing languages can change quickly. However, this is not to say that programming languages that are used in industry will always be the newer ones that offer newer features. Maeyerovich et al. [20] conducted a study analyzing open source projects and surveying 13,000 programmers in order to determine what makes programmers use certain languages. They found that open source libraries, existing code, and experience are the strongest factors that programmers will consider when choosing a language to develop in. Additionally, they found that while a small number of languages account for a majority of the language use, there is still a a large number of languages that are used to support more niche user bases. Schools cannot keep up with the constant changing of the workplace or address the wide variety that can be found in the work place as it is impractical. Due to this some schools may be teaching languages and skills that are out of date or no longer needed in the software engineering

workplace. A new graduate may find that the environment of working in the real world is quite different compared to university experiences even when looking at capstone courses [1].

Gaps that are correlated to tools and programming languages may not correlate with more critical issues within software engineering education such as, what the proper teaching methods for different software engineering courses are [25], or what the strengths and weaknesses are of online versus in person programming courses are [7]. However, understanding these gaps could help us narrow down some sources of a larger issue that may exist within software engineering education. The larger issue being whether or not current software engineering education provides students an adequate understanding of fundamentals like good testing practices, understanding the maintenance of an application, or understanding how to properly develop a more large-scale application.

The goal of my work is to find a novel way to close the gaps in software engineering education that are created by the constant evolution of the software engineering field. To help identify the existing gaps, I surveyed and interviewed industry professionals. With the knowledge of the gaps I developed a platform named "SE Gap Awareness" and recruited students to use the platform so that they can not only learn about the gaps that industry professionals think exist but they can also explore resources related to those gaps. The results show that students are not only interested in learning about the gaps that industry feels they have, but that students are interested in exploring resources related to different skills even if they are not directly correlated with an identified gap.

To reach a solution that attempts to aid in the expansion of students' skill-sets that they obtain from software engineering education by exposing them to tools and skills that may be under-taught or not taught at all, the following research questions were developed.

RQ1: What gaps do professional software developers think exist in software engineering education?

RQ2: What gaps do students think exist in their education?

RQ3: How can we close these gaps without causing large administrative overhaul (new course creation, course redesign, etc.) educational programs?

Chapter 2 discuses related works. Works that verify the disconnect between what industry uses and what students learn are first mentioned. Then works that examine previous attempts at identifying gaps through similar methods used in this study are discussed. Following that, works that discuss ways that gaps are potentially being addressed are mentioned. Chapter 3 provides details for the novelty and design of the study. Chapter 4 discusses the workforce survey and interviews and their results. Chapter 5 then discusses the "SE Gap Awareness Platform" design, study, and results. Chapter 6 discusses the results from the survey, interviews, and platform study. Chapter 7 discusses the limitations, threats to validity, and future work of this study. Chapter 8 discusses the conclusions that can be taken away from this study.

Chapter 2

Review of Literature

The study of gaps in Computer Science related education is not a new idea. Many studies have been conducted in this area. This area is important to continue to advance as Ghezzi et al. found that one of the biggest challenges that software engineering instructors face is finding a way to close the gap between what is taught in the classroom and what is useful in the real world [10]. Oguz et al.[24] conducted an intensive study into the gaps between industry and education and concluded that the the gaps were here to stay due to the fact that industry will continue to innovate.

Related works focus on how the gaps are not strictly related to coding skills. A study conducted by Groeneveld et al.[13] finds that communication is a vital skill to being a successful software engineer that may be frequently overlooked in software engineering education. Ng et al.[23] found that there is a large gap in the business social skills that are taught to software engineering students.

Garousi et al.[9] conducted a systematic literature review of papers relating to the gap between software engineering education and industry needs. They found that the importance of mathematical and engineering foundations are often overemphasized in software engineering programs. Kitchenham et al.[17] also found similar results using different survey methods. Groeneveld et al.[12] conducted a syllabi review of several undergraduate software

engineering courses that are taught in the EU. Through their syllabus review they found a general trend to an increase involvement of "soft skills" in course work. However, even though there may be an increase in involvement this does not guarantee that the increase in the skill is catching up to industry standards but it is a step in the right direction. Callahan et al.[4] conducted a study to attempt to make software engineering education focus more on professionalism. They created a whole new graduate course to attempt to address more industry needs without sacrificing the existing curricula.

2.1 Prior Survey and Interview Based Works

In a study conducted by Dianne Hagaan [14], employers of recent graduates were asked to provide suggestions for improvement for universities. Two interesting takeaways from this study are that the respondents felt that more work experience should be included in education and that students should receive more training in general communication skills. Lee et al.[19] surveyed students and recruiters to try determine if there is a gap between the skills that students find important and the skills that recruiters find important. Through their study they found industry in general is demanding more than students expect. These studies in particular can be helpful in answering RQ1 and RQ2. Some sample potential gaps that were identified in these studies for RQ1 and RQ2 are Object Oriented Languages, Documentation, and Communication. There is a general disagreement noted especially in the study conducted by Lee et al.[19] that shows that students do not think they need as many skills as industry thinks they do. The aforementioned study performed by Oguz et al. [24] reached their conclusion that gaps will require constant attention by surveying and interviewing recent new graduates and comparing those takeaways to software engineering job postings.

2.2 Addressing Gaps in SE Education

The works previously mentioned show that there is a potential disconnect between what industry wants and what students can offer them. However, this does not mean that there is no work that is attempting to address any gaps between what industry wants and what students are taught in software engineering education. Marisa Exter [6] found that the best way education currently succeeds in preparing students to work for industry is providing them with a strong foundational knowledge base that can be applied to learn new skills. This is important to note because changing how courses are structured is not the goal of this project. The goal is to maintain the foundation knowledge that students are being taught while expanding their skill-set to better address the more specific skills that industry desires.

Ellis et al. [15] successfully updated an existing CS2 course that mainly focuses on problem solving to include more technical skills. Additionally, Ellis et al. [15] concluded that the updating of this course could potentially lead to students exploring more resources outside of the classroom.

Different guidelines have been developed in order to ensure that software engineering students are being taught an effective skill-set. However, Moreno et al.[21] conducted a study analyzing the 2004 Software Engineering Curriculum Guidelines for Undergraduate Programs and the Curriculum Guidelines for Graduate Degree Programs in 2009. With these guidelines they compared them to three different industry profiles and found that none of the guidelines created were able to meet any of the industry profiles' requirements. However, more current guidelines are attempting to address the weaknesses of previous guidelines. The ACM Guidelines for 2020 [8] discuss how the previous guidelines that they released in 2005 create a "skills gap" and they discuss their newer approach to the guidelines which they call

"Competency-based Computing Education". This newer approach focuses on using "competencies" by defining different skill areas within software engineering and then different skills within those areas. These areas follow skills such as software design, software testing, software security, etc. There are many skills that the guidelines mention but none of the guidelines list out specific tools or programming languages. Additionally, Walter Schilling [26] noted that the 2014 IEEE Guidelines may be behind and recommends a new guideline that includes integrating DevOps into software engineering education.

In order to effectively build on the existing structure of coursework to help students obtain a broader skill-set, perhaps more non-traditional methods of education could be beneficial to ensure that the students being taught newer skills in an effective manner. One such non-traditional method that could help is using games to aid in students learning. A systematic literature review conducted by Kosa et al.[18] found that game-based learning can enhance positive learning experiences for the students and can even improve the level of understanding by the students.

Another interesting way to potentially expand the skill-set that students obtain through coursework is to explore the concept of nudge theory to potentially change student behavior in the classroom. Brown et al.[3] studied applying nudge theory in undergraduate computer science courses to get them to conduct better software engineering behaviors. They accomplished this by creating a bot that is designed to nudge students in the right direction without directly rewarding them. Their work is built upon the idea of "digital nudging" which is defined by Weinmann et al.[28] as "the use of user-interface design elements to guide people's behavior in digital choice environments.".

A study done by Murphy et al.[22] found that providing students with forms of self-examination

such as practice exams improves a student's ability to comprehend the topics. Along these same lines my study aims to provide an opportunity for students to conduct their own forms of self-enhancement through guided resources that the student can optionally explore. The idea of students turning to online resources is analyzed in a study by Hao et al. [26]. In their study Hao et al. [26] looks at how students seek help online and they found that students mostly seek help online by searching online resources.

Chapter 3

Study Novelty and Design

Some existing studies recommend courses restructuring or even request new courses be added [2, 4]. My goal is to create an approach that will help close the existing gaps without causing so much administrative overhaul.

All previously mentioned studies either do not directly interact with students to address the gaps or they do so in a manner that more closely correlates to a classroom session. Most of the aforementioned studies will talk about gaps by interviewing or surveying industry professionals [6], interviewing or surveying recent new graduates [24], or by looking at posted job requirements [21, 24].

Related work fails to address the existing gaps without causing big administrative changes. Some work recommends entire new courses be implemented. My work is different because it aims to close these gaps without causing new courses to be designed or for existing courses to be redesigned. My work aims to cause as little responsibility on the teacher as possible so that they can continue to focus on teaching the course without worrying about following every industry standard. The concepts should be what shines in coursework and through my work I hope to show that students are interested in learning more if they are simply nudged in the right direction. An example of the kind of flexible knowledge that students can be provided is the website GeeksForGeeks. GeeksForGeeks provides a wide variety of

tutorials for different languages but also provides tutorials and descriptions for topics that are not language specific like data structures, machine learning, algorithms, etc.

Chapter 4

Workforce Survey and Interviews

4.1 Workforce Survey

To help further identify what gaps may exist, I created a survey and distributed it to professional software engineers. The survey was distributed through personal professional contacts of the main author, the main advisor, and a second researcher. In addition to this, recruitment tweets were sent out via Twitter by the main advisor. At the end of the survey, the participants had the option to opt in to participate in an interview where they were given the opportunity to further elaborate on any of their answers that they gave in the survey. The survey asked the respondents to answer Likert scale questions for each of the skills shown in Table 4.1, where the scale is grading the new graduate hires that the respondent has worked with. The coding skills were determined by looking into the most popular coding languages as noted by the TIOBE Index in addition to including coding skills that the main author encountered in their professional experience. The Soft and hard skills were determined based on skills that were mentioned in prior works as well as skills that the main author found important in their professional experience. Each Likert scale question was a scale from 0 - 5 where 0 meant that the skill was not applicable in the respondents experience, 1 meant "not proficient", and 5 meant "very proficient". Additionally, after each Likert scale question, the respondents were given the opportunity to fill out a short answer question where they could elaborate on their Likert scale answers. The full survey can be found in section A.2.

Table 4.1: Skills by Category

Soft Skills	Hard Skills	Coding Skills
Communication	Github	C
Planning	Agile Development Practices	Java
Writing	Documentation	JavaScript
Requirement Scoping	Project Planning Tools	TypeScript
Teamwork	Docker (or similar build tools)	C++
	Cloud Development Tools (AWS, Microsoft Azure, etc.)	Linux
	Unit Testing	SQL
		HTML/CSS
		C#
		Python

4.1.1 Survey Results

A total of eleven professional software developers responded to the survey. The respondents worked for several different companies including Amazon, Fidelity Investments, VMware, ByteDance, Two Six Technologies, and ZS Associates. The respondents reported a variety of positions including Research Director, Software Engineer I-II, and Senior Member of Technical Staff. The average years of experience of the respondents was 9 years. The range of respondents experience went from 1 year to 32 years so a broad variety of experiences were represented in these results with most of the participants falling under 10 years of total experience. Figure 4.1, Figure 4.2, and Figure 4.3 show the results of the Likert scale questions that were asked. Averages were calculated with 0's excluded. However, several skills did have high 0 counts. There were only a few 0 responses for the hard and soft skills but the coding skills received a larger number of 0 responses and are shown in Figure 4.4. Additionally, there were very few answers given for the short answer elaboration questions. From the short answer elaboration questions soft skills were determined as weak for new hires, and one respondent pointed out that most new hires are not familiar with cloud development tools. Overall there were very few elaboration answers given.

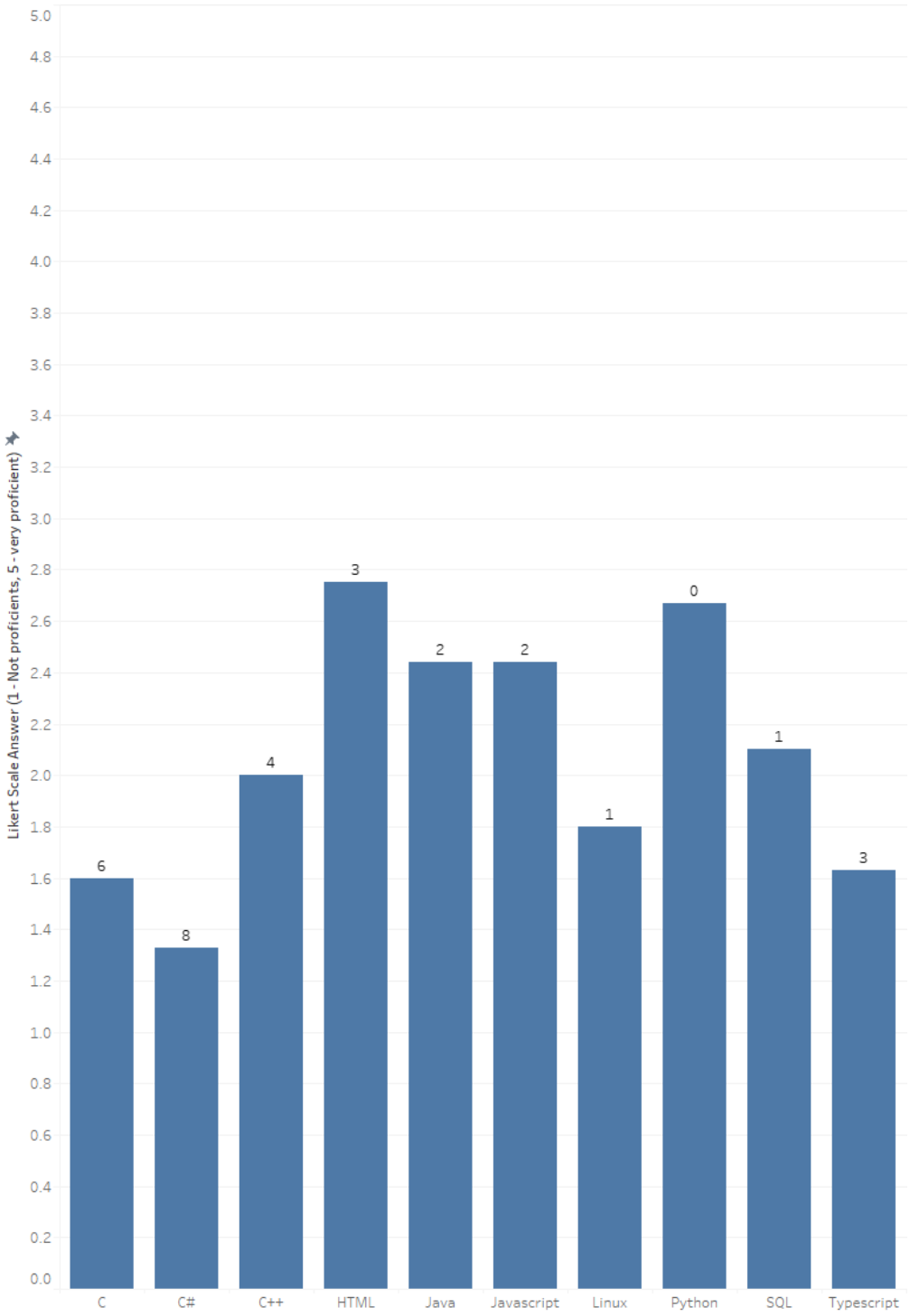


Figure 4.1: Survey averages for each coding skill from 11 respondents, responses that answered 0 are removed from the averages, the number displayed above the bar represents the amount of responses that answered with "0" which meant N/A

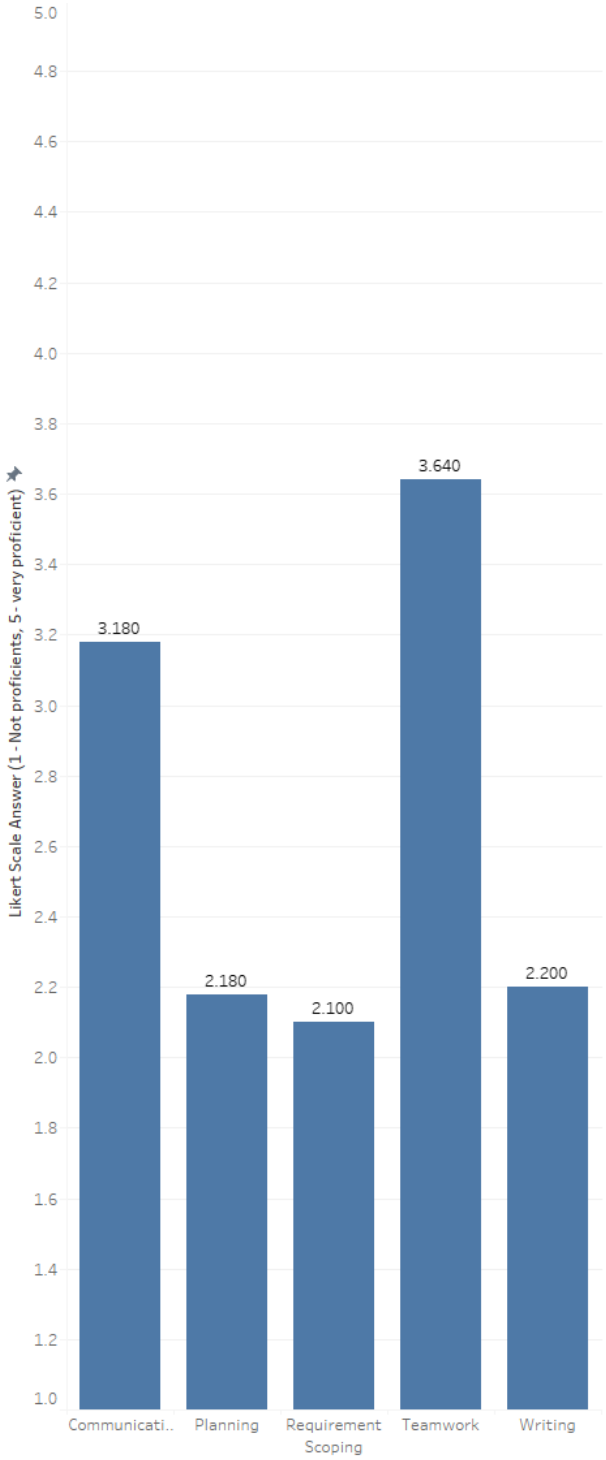


Figure 4.2: Survey averages for each soft skill from 11 respondents, responses that answered 0 are removed from the averages

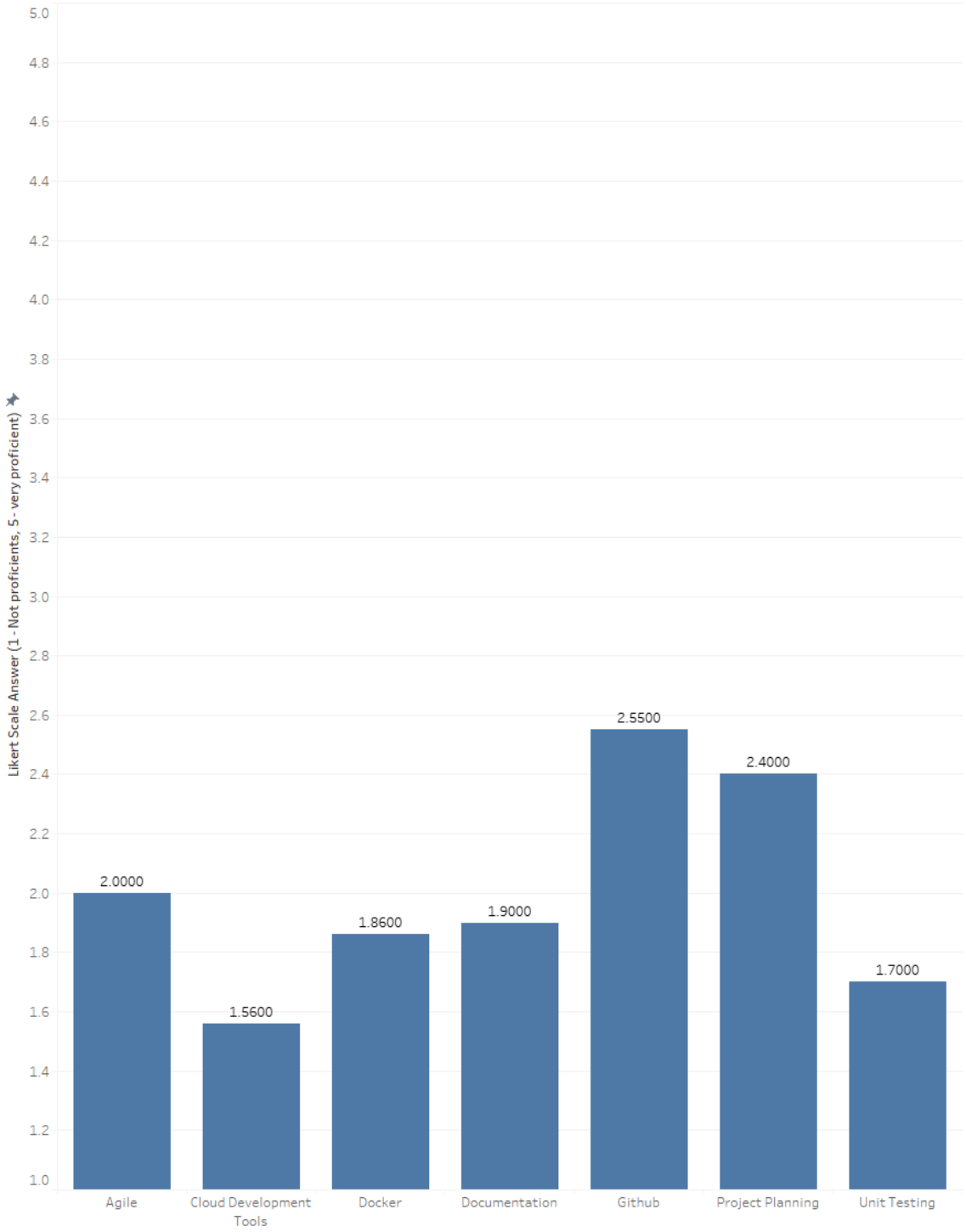


Figure 4.3: Survey averages for each hard skill from 11 respondents, responses that answered 0 are removed from the averages

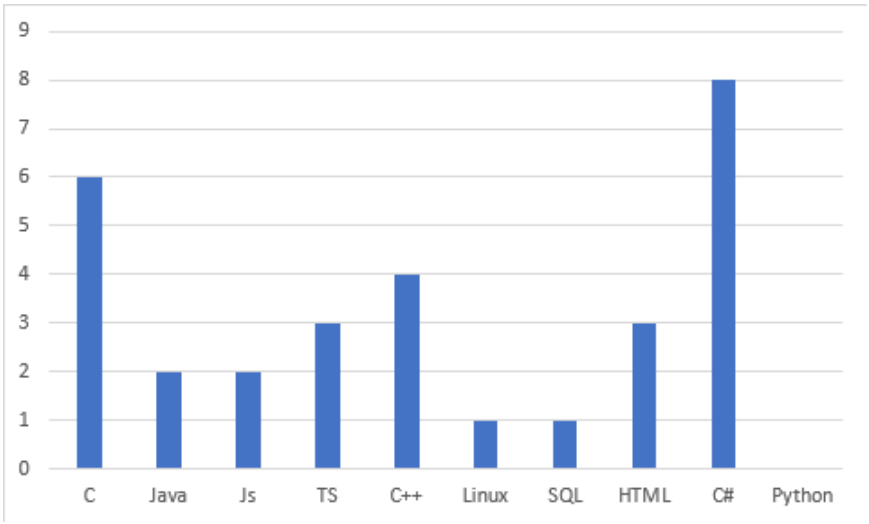


Figure 4.4: Count of 0 Answers for Coding Skills from 11 respondents

4.2 Workforce Interviews

At the end of the survey, the participants were given the opportunity to provide contact information in order to setup an interview. The interviews were semi-structured and the main goal of the interview was to allow the participant to elaborate further on any response they gave during the survey. The interviews were all recorded and transcribed and then analyzed to derive common themes from each interview. The interviews were analyzed by two researchers looking at the comments made by the participants to derive themes of skills new hires lack based on employer responses.

4.2.1 Interview Results

Three survey respondents agreed to an interview to further elaborate on the questions they were asked in the survey. All of the interviews were recorded and transcribed. Once all of the interviews were complete, the transcripts were analyzed through an open coding process. Through this process myself and another researcher assigned themes to the answers given in the survey and then discussed them until we reached an agreement on the the important takeaways from the interviews. Two main themes were extracted from this process.

Theme 1 Soft skills are weak for new graduates.
Each interviewee talked about how new graduates will start off with poor soft skills but they will improve them over time. Specifically, each interviewee mentioned that communication is poor for new hires. This lines up with several other studies that were discussed in the related works section. This also lines up with several responses from the elaboration portion of the survey.

Theme 2 Ability to learn a new language is important.

Each interviewee mentioned that there can never really be one language to focus on. They talk about how its important for new graduates to be able to take their existing coding skills and apply them to different languages as needed.

From these themes we can derive some initial conclusions for RQ1. While Figure 4.2 shows that three weakest soft skills are Planning, Requirement Scoping, and Writing, the interviews lead us to believe that soft skills overall can be viewed as a gap. Additionally, the interviews showed us that focusing on a specific language is not a large concern but that is more important that a new professional is efficient at learning a new programming language. This second theme may also be an insight into RQ3.

Chapter 5

SE Gap Awareness Platform

The goal of this platform is to provide a tool that students can use to help broaden their skill-sets. The students will be given the opportunity to answer the Likert scale questions that were asked of industry except this time the students will be grading themselves. Students will also be able to play a guessing game for each of the skill categories to see if they can guess which skills industry thinks recent new graduates are worse at and which ones industry thinks recent new graduates are better at. The orderings that students will guess at are pulled from the data shown in Figure 4.1, Figure 4.2, and Figure 4.3.

5.1 Platform Design

The SE Gap Awareness platform was developed using Django Python combined with PostgreSQL and deployed the platform on Heroku. All of the interfaces were wire-framed and reviewed by the main advisor to ensure that the user interface was acceptable. Almost all interactions with the platform were stored for tracking purposes. The interactions that were recorded include, Likert scale answers, guessing game attempts, and resource link clicks. When students first visit the platform they see the home page of the platform that includes a section for them to log in. The students cannot interact with the platform without logging in. Once the students are logged in they gain access to the "Game" tab and the "Resource" tab, Figure 5.1 shows the home page once a student has logged in. Once students open up

the "Game" tab they are first asked to answer a series of three questions that are extremely similar to three of the questions that professional software engineers answered in the survey discussed earlier. These were Likert scale questions that asked the students to self evaluate on how they view their proficiency in different skills and can be seen in Figure 5.2. The results of these three questions will be used to answer RQ2.

Following the three Likert scale questions, the students are met with three more questions where each is a simple guessing game where the students try to guess the correct order of skill proficiency that industry professionals say recent new graduates have. A sample guessing game question is shown in Figure 5.3. The Guessing Game is a simple drag and drop ordering for the skills. The students could then click a button to check their answers and skills that were in the correct location were highlighted green and skills that were in the incorrect location were highlighted red as shown in Figure 5.4. The purpose of the guessing game is to determine if students are interested in learning how industry ranks certain skills and then to see students will be more interested in exploring resources related to the lower rated skills. In total there are three Likert scale questions and three guessing game questions. Once the students were finished with their three guessing game questions, they are then prompted to move onto the resources tab, shown in Figure 5.5, to explore resources related to the skills mentioned throughout the game. The resources tab contained a collection of resource links that were categorized by skill.

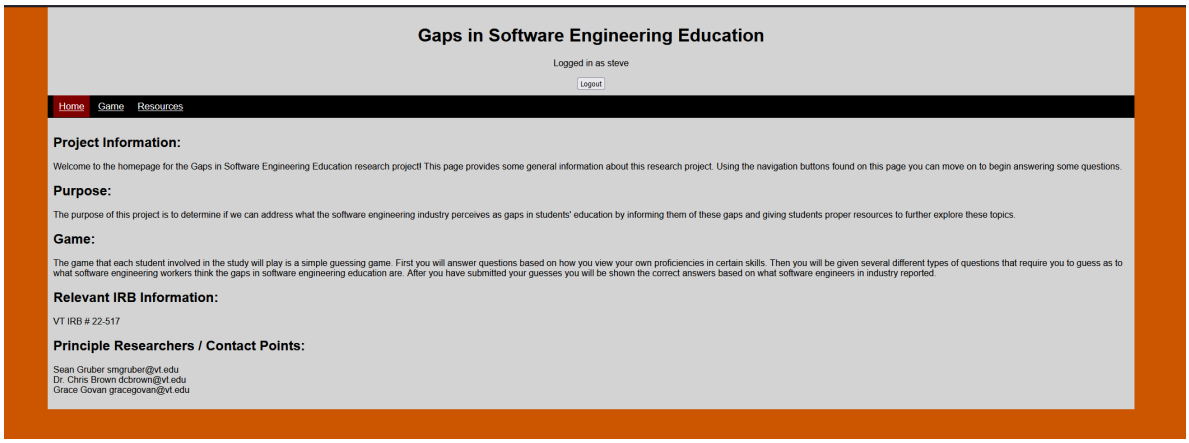


Figure 5.1: Home screen of the Platform

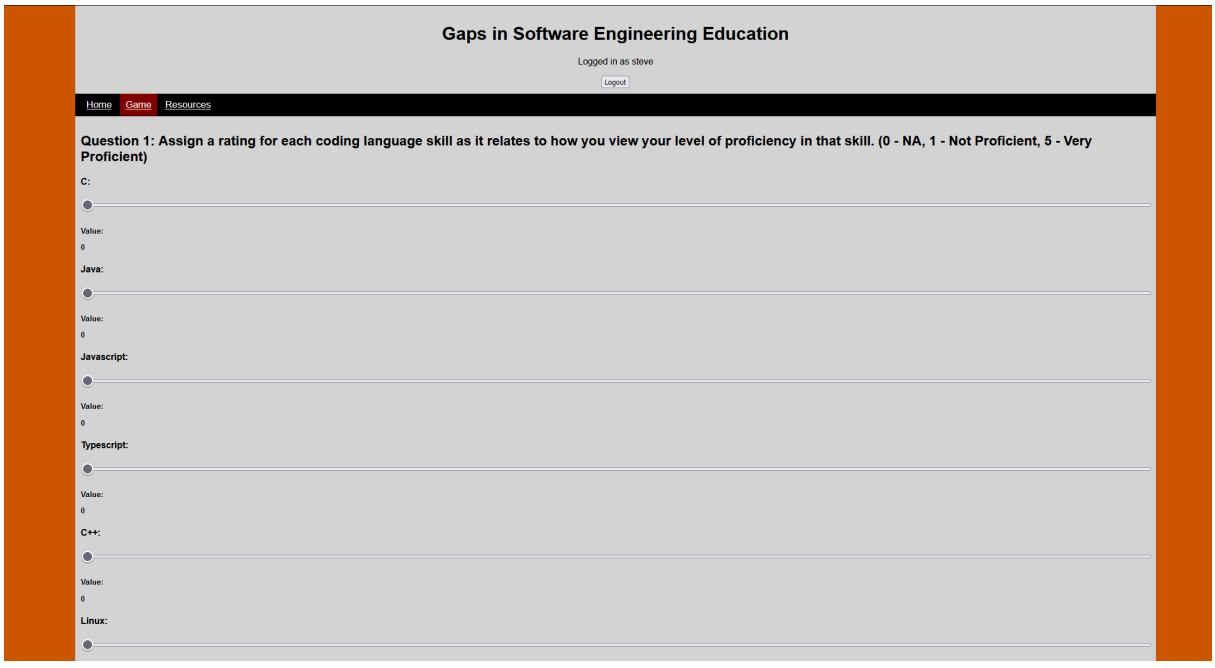


Figure 5.2: Likert Scale question for coding skills

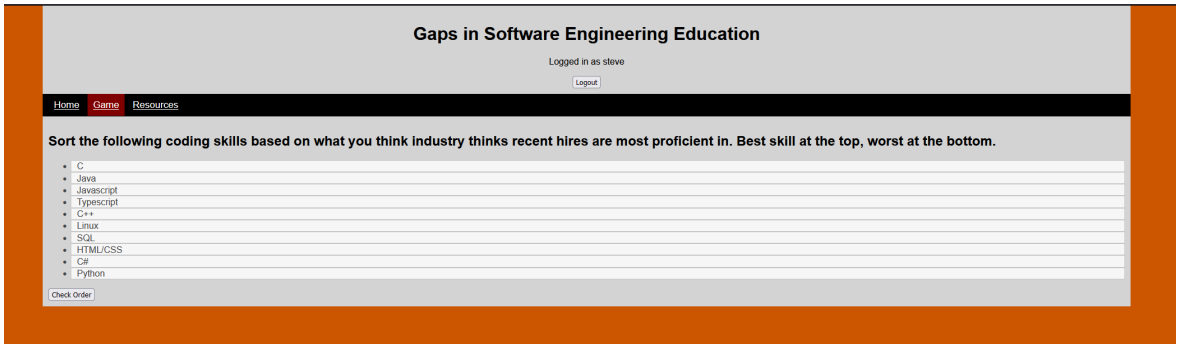


Figure 5.3: Guessing game question

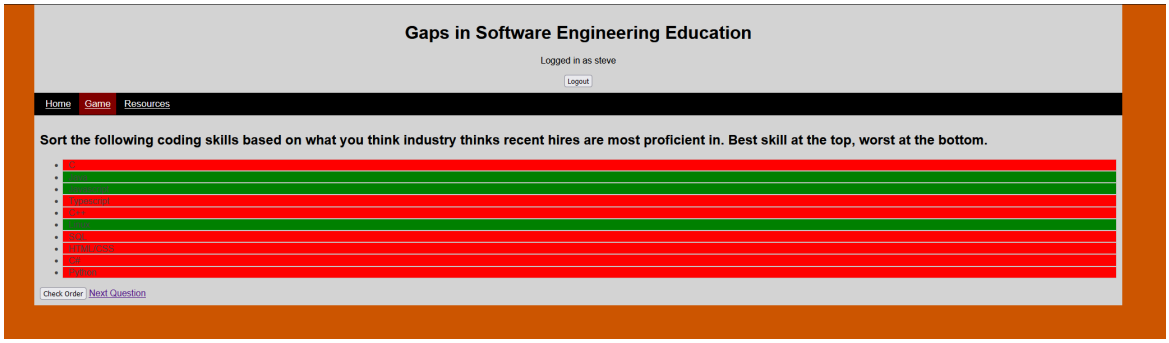


Figure 5.4: Guessing game question where a guess has been submitted.

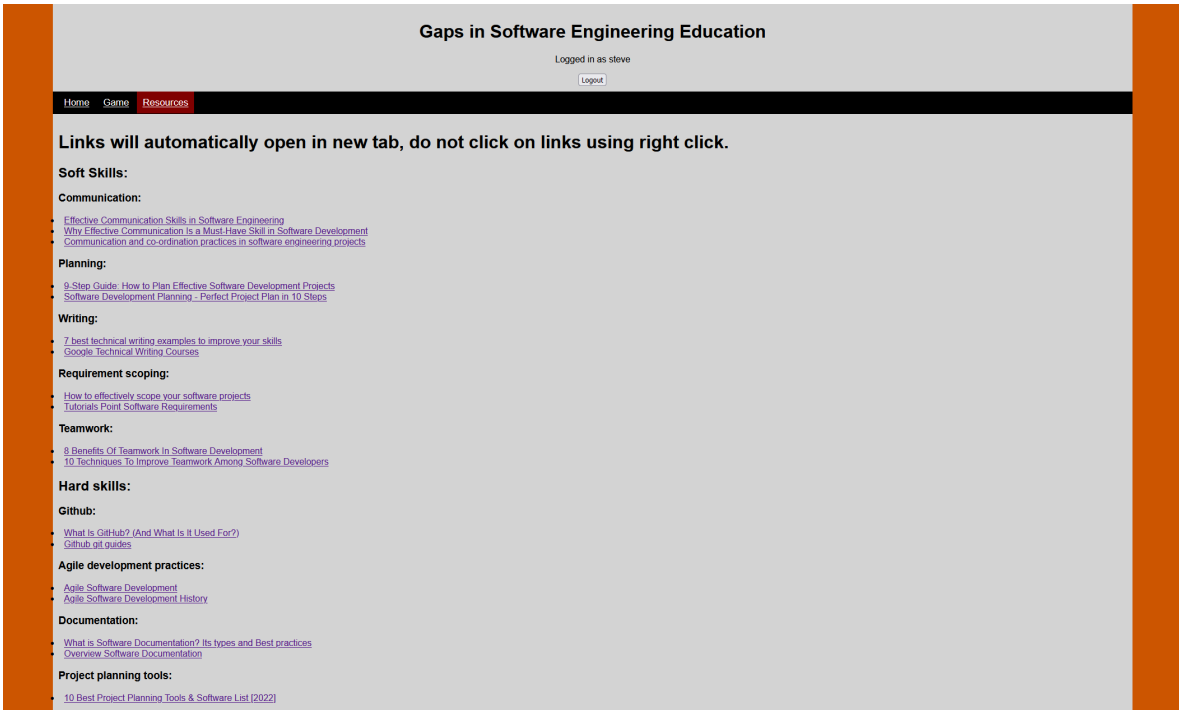


Figure 5.5: Resources Page

5.2 Platform Study

To conduct the user study on students I reached out to different professors who were teaching software engineering courses and asked if I could run the study in their class. If the professor agreed to participate I went to their class and presented the platform to the students. Four sections of CS-3704, an undergraduate software engineering course, and one section of CS5704, a graduate-level SE course participated in the study. The presentation took less than five minutes of class time and students signed up to participate in the study by filling out a consent form. Once the consent form was filled out I provided the student with a login for the platform via email. All of the student's interactions with the platform were anonymous and I recorded any interactions with the resource section of the platform. The following interactions were recorded: resource links clicked, Likert scale question responses, and guessing game question responses. Students were given access to the platform for around a month. After a month the participating students were sent a retrospective survey to get their thoughts on the platform to gauge their interaction with the platform.

5.3 Platform Interaction Results

The study was run twice over two semesters. Each semester followed the same procedure that was described in the previous section. Over both semesters 112 students signed up to participate. Out of those 112 students, 61 interacted with the platform in some manner. Figure 5.6, Figure 5.7, Figure 5.6 show the results of the Likert scale questions that the students were asked. These questions are the same Likert scale questions that were asked in the survey that was sent out to industry professionals. As with the industry survey very few 0 responses were received for the hard and soft skills but the coding skills had quite a few

0's as shown in Figure 5.9. The guessing game had a total of 637 recorded attempts. Out of the 637 recorded attempts 60 of those attempts got all of the skills ordered correctly. Out of all of the students that attempted the guessing game only 20 out of the 61 students got at least one guessing game question correct.

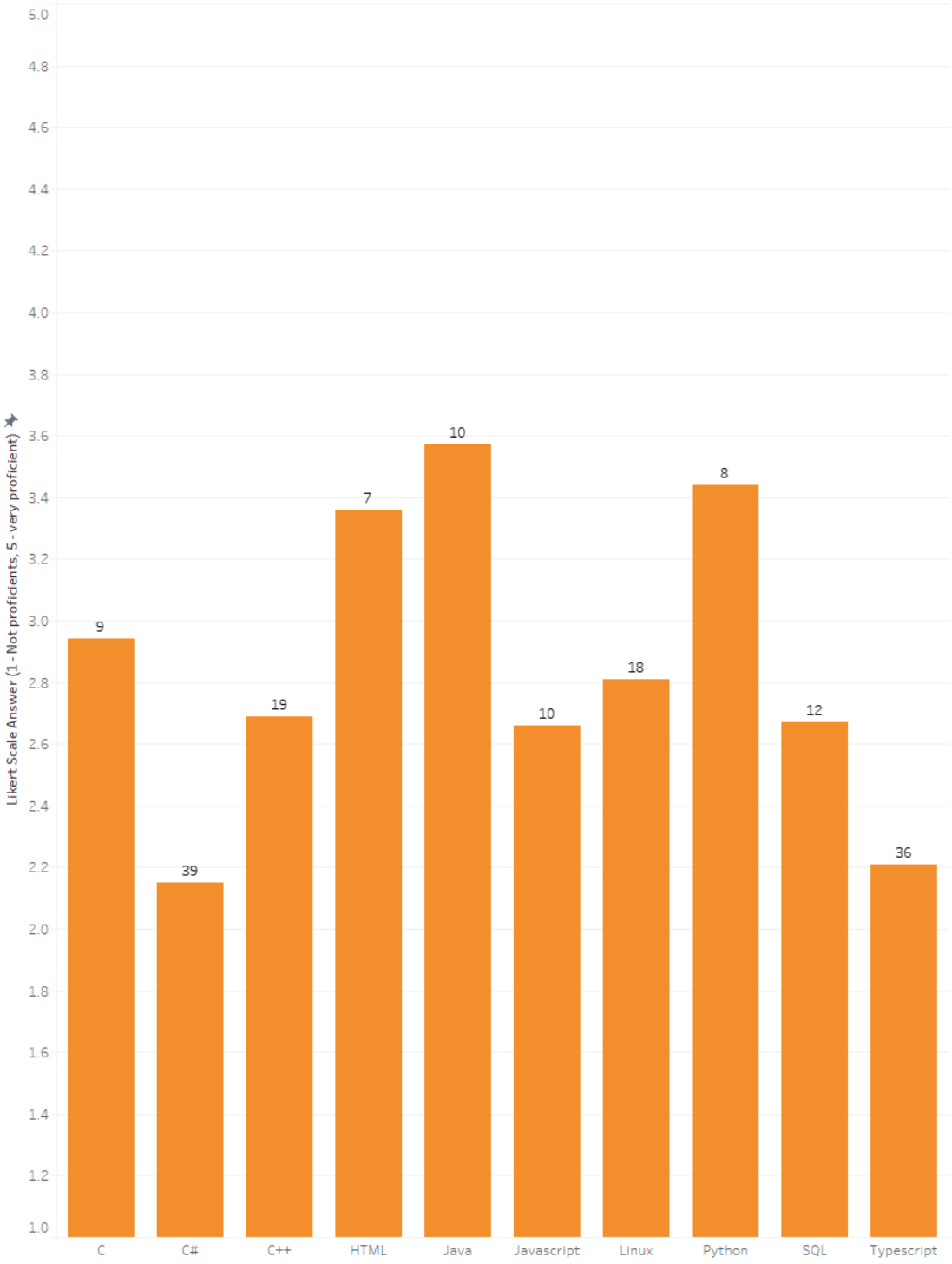


Figure 5.6: Survey averages for each coding skill, responses that answered 0 are removed from the averages, the number displayed above the bar represents the amount of responses that answered with "0" which meant N/A

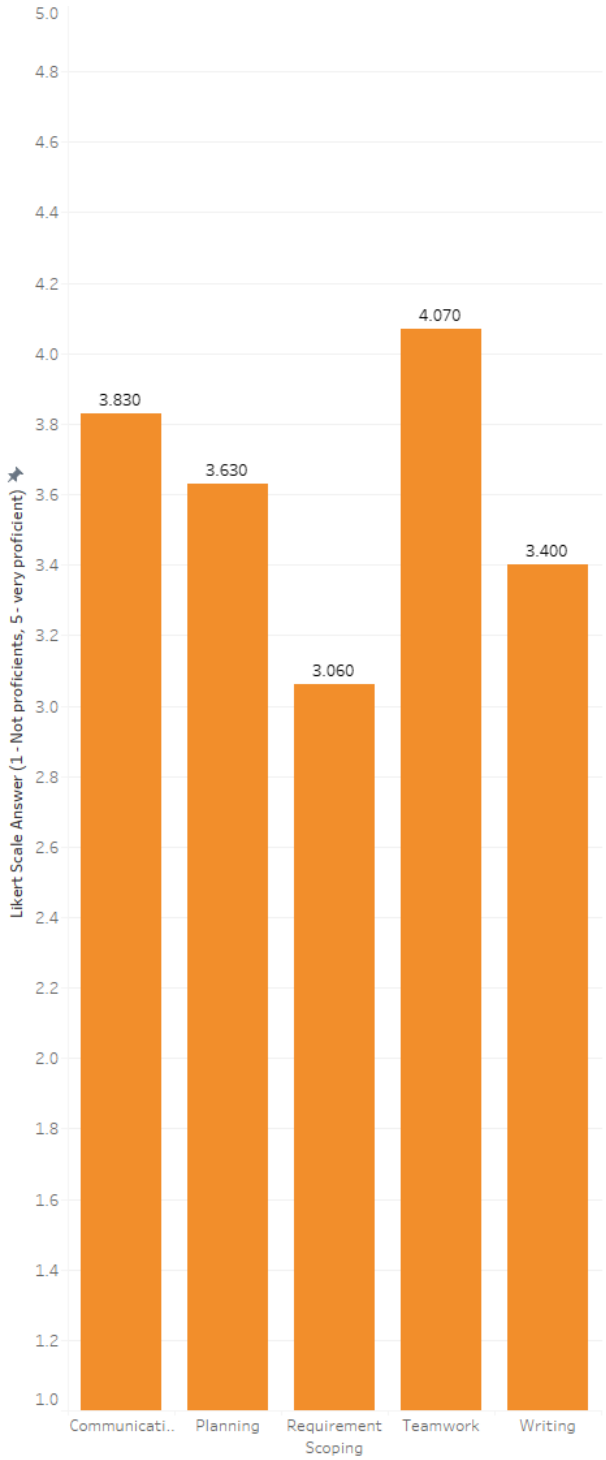


Figure 5.7: Survey averages for each soft skill, responses that answered 0 are removed from the averages

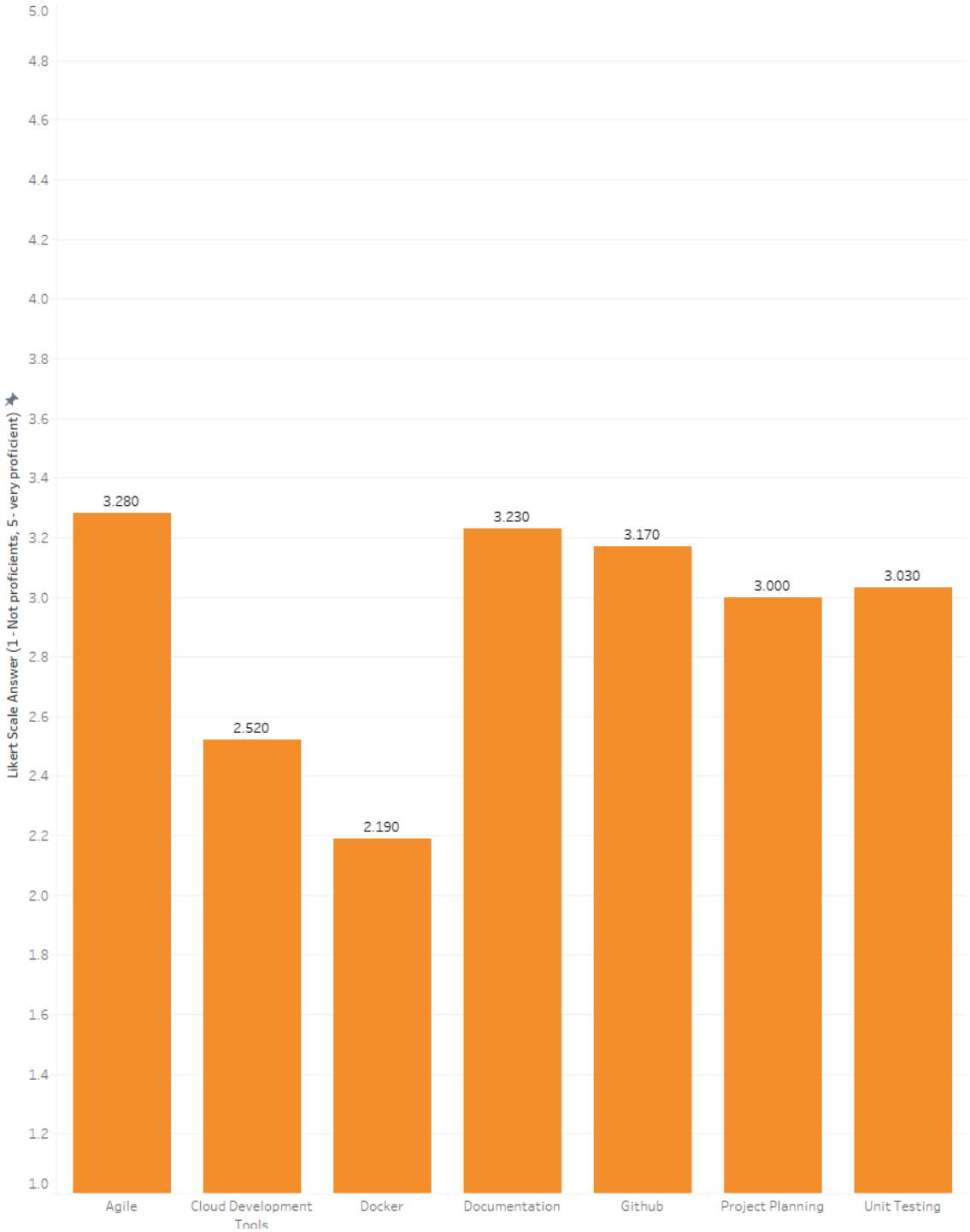


Figure 5.8: Survey averages for each hard skill, responses that answered 0 are removed from the averages

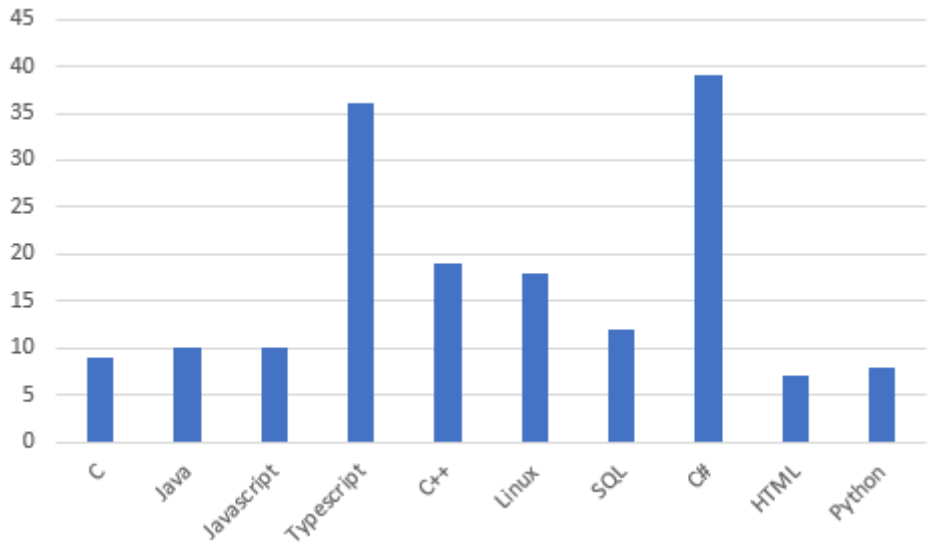


Figure 5.9: Zero Count for Each Coding Skill

Out of the 61 students that interacted in the platform, 33 students explored at least one resource from the resources page. Figure 5.10 shows how many times a resource was explored related to an individual skill. Communication skills were by far the most practiced skill. Additionally, each skill was practiced at least once.

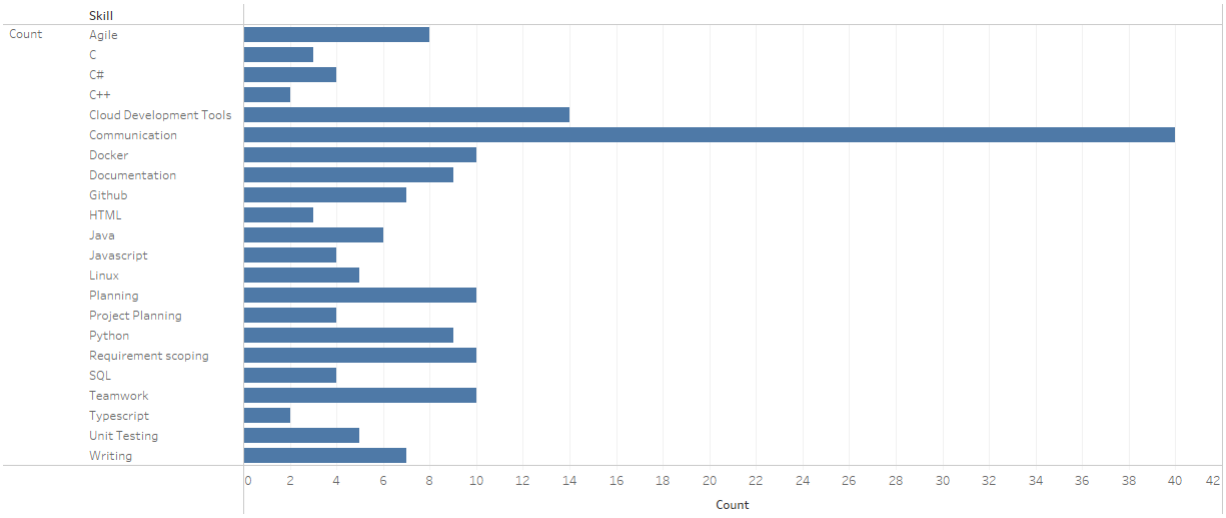


Figure 5.10: Times a resource was explored for each skill

5.4 Retrospective Survey Results

The retrospective survey was sent out to learn how the platform could be improved. Students were asked questions about how much they interacted with the platform, if they found anything surprising, and were asked to suggest improvements. Out of the 112 students that the retrospective survey was sent to, 31 students responded. Out of the 31 students that responded 18 of them said they used some of the resource links that were provided. 10 students said that they were surprised about the gaps that were presented in the guessing game. This helps in part to answer RQ2 as we see that there is a disconnect between what students think they are good at and what industry thinks they are good at.

Chapter 6

Discussion

6.1 Likert Scale Discussion

Figure 6.1, Figure 6.3, Figure 6.2 show a comparison of the Likert scale question answers given by students and by industry software engineers. Keep in mind that the industry answers are evaluating recent new graduates and the student answers are evaluating themselves. Something interesting to note here is that for all of the skills across all three categories, the students rated themselves higher than industry rated them. This may show a need for students to have greater access to tools and resources as they may not be aware of how much they do not know. On top of students rating themselves higher than industry they also simply rated themselves high on the Likert scale. This could potentially be due to confusion on the Likert scale questions which comes down to poor question design.

However, with those takeaways in mind several key factors are important to note for mainly the coding skills. First, the representation of answers for the coding skills are smaller than the overall answers for the both surveys as a whole due to the fact that not every industry respondent will have experience with each language list and not every student will have experience with each language listed. This is most likely the cause of the high amount of 0 answers that some of the coding skills obtained. When considering the already low amount of respondents, no statistical significance can be pulled from these data sets. Additionally,

there are direct conflicts in the data between the survey and interview results. Soft skills being overall weak is not represented at all in the survey but was the main talking point in the interviews. The same can also be said for the student Likert scale response which again rate the soft skills relatively higher.

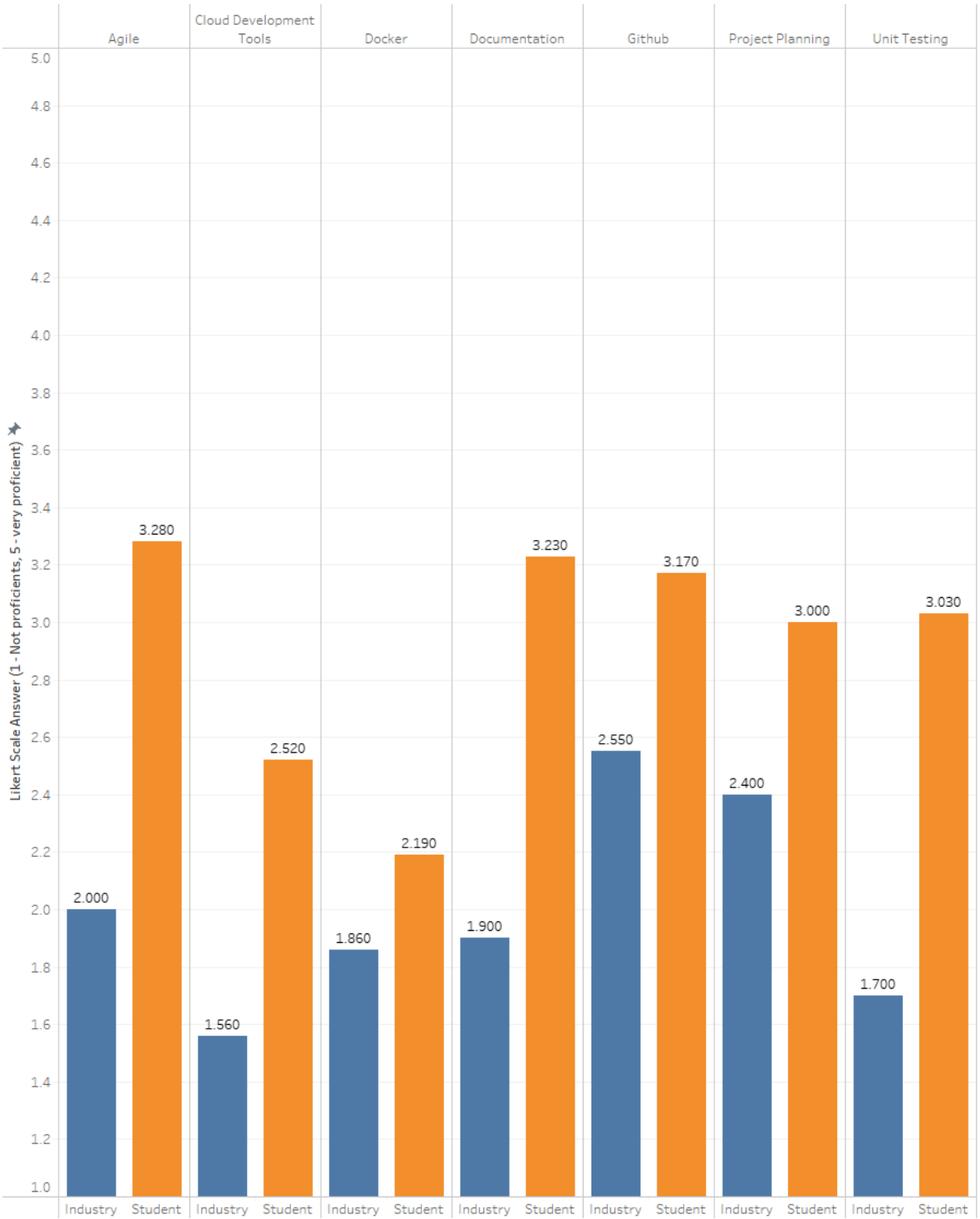


Figure 6.1: Comparison of Likert Scale Answers from Industry and Students for Hard Skills

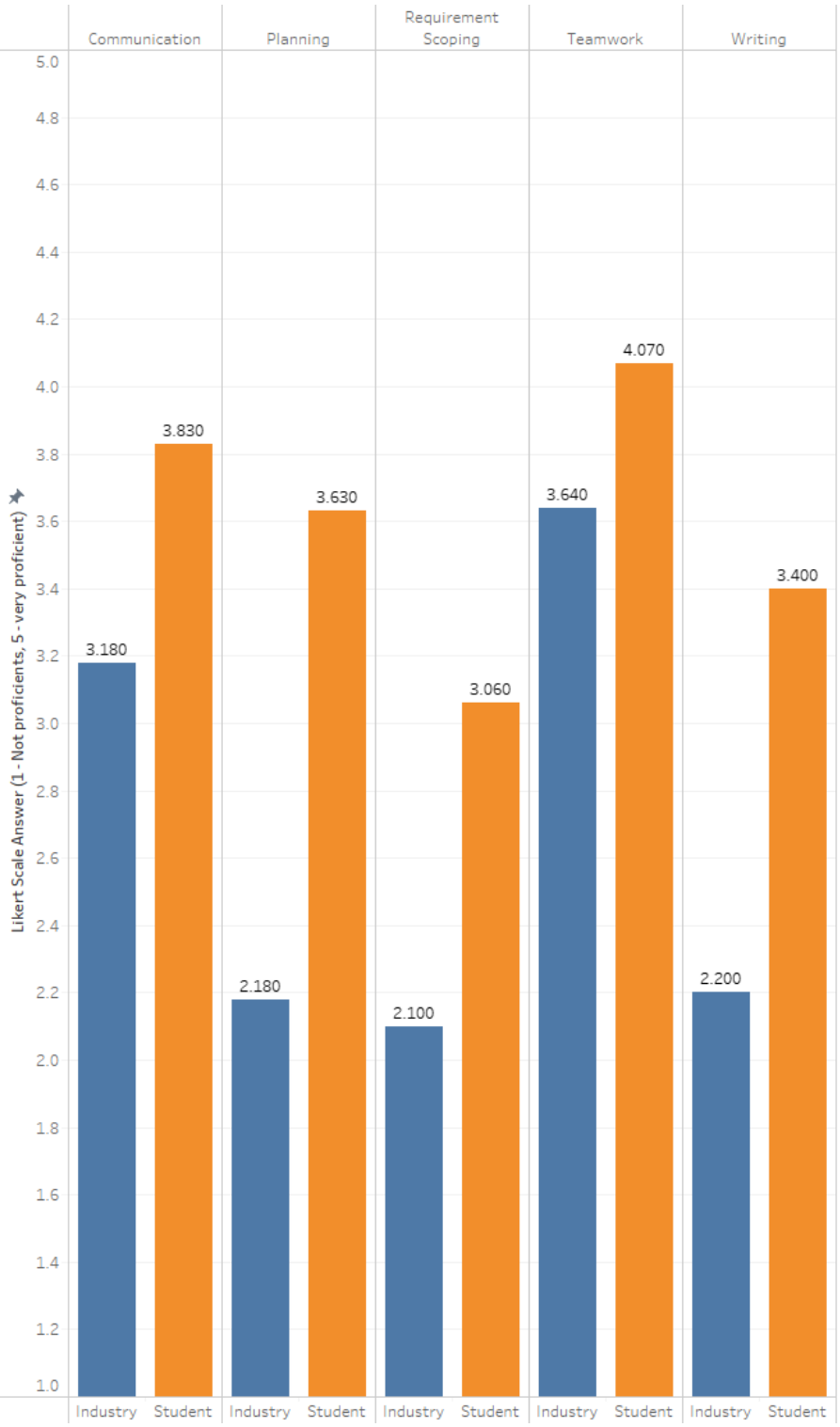


Figure 6.2: Comparison of Likert Scale Answers from Industry and Students for Soft Skills

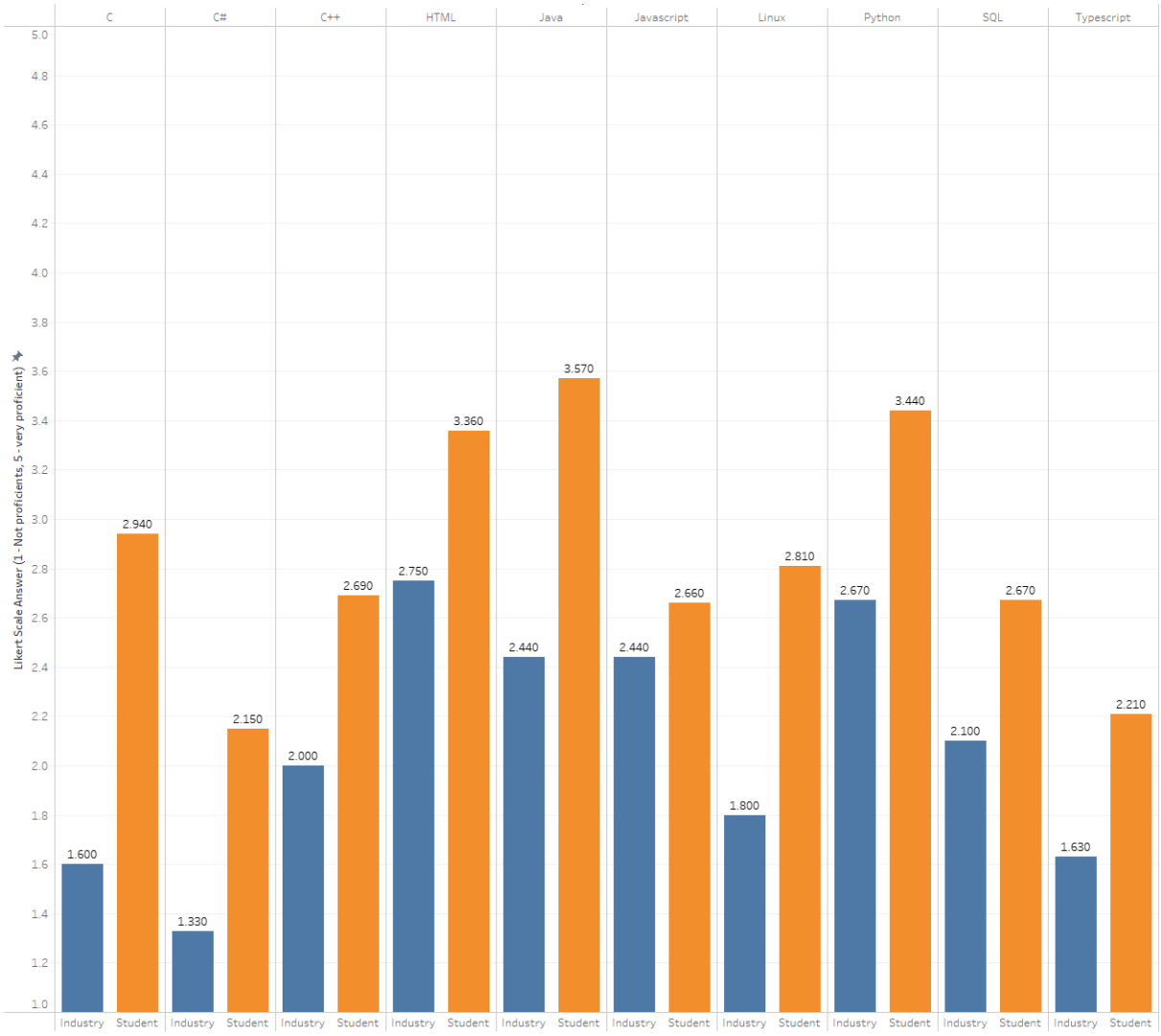


Figure 6.3: Comparison of Likert Scale Answers from Industry and Students for Coding Skills

6.2 Platform Interaction Discussion

The high interaction with communication resources show in Figure 5.10 is interesting. While soft skills like communication were frequently brought up in interviews and related works, based on the survey the communication skill was rated the second highest proficiency according to the surveyed industry professionals. This means that when the students played the guessing game they saw that industry was giving them relatively high scores when compared to the other soft skills. However, despite this students were still drawn towards the resources related towards this skill. This could be attributed to several factors. For one, the communication section of the resources page at the top of the page. This could mean that students were only really interested in exploring a few resources and just clicked on the first one. Another possible explanation is that the students feel that they could benefit the most from improved communication skills or perhaps the students were just curious what a communication resources looked like. While students may not have graded their own communication skills on the lower end, they may have felt that the communication resources would lead to the greatest direct benefit for them. Students also explored resources related to cloud development tools on a larger scale than the other skills. This seemed to be an area of great interest to students as well as in the retrospective survey one student had even requested for other cloud tools and development frameworks to be included in the resources section to allow for further exploration. Overall, students did not show a great deal of interest in exploring the skills related to any coding skill.

Chapter 7

Threats and Future Work

While this study aimed to achieve an understanding of education gaps from industry professionals, only 11 industry professionals responded to the survey and only three interviews were conducted. Due to this smaller sample size the results are not generalizable to all employers and some gaps may remain uncovered. However, the platform was designed with this in mind. The platform included resources for every skill that was listed, not just the ones that were ranked poorly by industry. Additionally, the only student participants are from Virginia Tech and in the future could benefit from involvement with other universities. Additionally, the industry respondents were grading recent new graduates but we do not know where these new graduates came from.

In the future there are many ways to expand this work. More interaction with nudge theory would be beneficial. For example, there has been other studies done on a badge system not just in the workplace but in the classroom [11]. If some kind of leader board existed for say the most resource exploration perhaps this could have driven even more interaction with the platform.

Additionally, some work into small benefits for the students for using this platform is an interesting avenue. For example, many programs use some form of an online submission program for coding assignments. In some cases the amount of submissions per hour that a

student can make are limited, but if interaction with the platform awarded them one more submission then this could be a guiding benefit that overall positively impacts the student.

The platform could potentially be even further expanded to allow for more direct student to student interaction. For example, students could potentially submit the resources that they find to the platform and receive some form of credit for it. The students could also rank existing and submitted resources within the page to let other students know if they found certain resources more useful than others. This could encourage more online resource exploration by the students. A study conducted by Hooshangi et al. [16] found that the early exposure to the practical skills led to an increase in personal project development from students. As students are exposed to more skills through the platform then potentially they will increase their own knowledge even further with personal projects.

To increase gamification of the platform, each newer coding language could have some form of tutorial that provides students with the basics of a new language within the platform.

The aforementioned paper from Walter Schilling [26] also has more insights that could be useful for advancing this work. The idea of the DevOps course that is being discussed in the paper came from a committee of members from industry that were recommending new skills that should be taught in education. Additionally, the committee was closely involved in the design of the course to ensure that the concept of DevOps is being taught the same way it is used in industry. Perhaps forming a similar committee to look over the resources of the platform could be beneficial as well as the committee could further help to narrow down what skills students are missing out on in the education.

A long-term study would be beneficial to understanding the impact of SE Gap Awareness

resources and motivate future tools/resources for students. Through a long-term study students could potentially be evaluated on different skills and then after being given a longer time period with the platform they could be re-evaluated to see if their knowledge of the skills improved. Unfortunately tracking how the gaps are more broadly impacted would be difficult as we would have to somehow follow a student that interacted with the platform into a workplace and then receive a coworker's evaluation of them that way. There are too many factors that could influence a study like that to in the end be able to claim that the platform was what led to potentially closing the gaps.

Overall, the possibilities with a platform like this are limitless. The most important piece to keep in mind is that the platform is meant to exist for the students benefit so that they can have more successful careers.

Chapter 8

Conclusions

The goal of this study was to find a novel way to address existing gaps in software engineering education without causing massive overhaul of existing courses. The platform that was developed succeeds in providing students with a variety of different resources related to a broad range of skills. While only a subset of the students interacted with the platform, many students did at least explore one resource. The goal of the guessing game was to drive student interaction to allow them explore what industry says the gaps in their education are. After reporting more than 600 attempted guesses from a total of 60 students, it seems that this goal was met.

RQ1 itself cannot be answered from the data presented in this study as the collected data is too small to be significant and contains some conflicts with the interview themes. Even with that in mind the following insights can be taken away as preliminary work into answering RQ1. Through the survey, an initial ranking of skills in terms of new graduate proficiency was achieved. The gaps are identified as the lower rated skills within each category. This leads to the following skills being defined as gaps: Planning, Writing, Requirement Scoping, Cloud Development Tools, Unit Testing, C#, C, and Typescript. Through the interviews the most common takeaway was that soft skills like communication are extremely important for new graduates. While communication is not ranked low, its identification as a gap came through the interviews.

Again RQ2 itself cannot be answered by the results of this study however, some insights can be taken away as preliminary work into RQ2. Research question 2 is answered by the Likert scale questions asked of the students on the platform. Again here the lower rated skills are meant to be identified as the gaps that students feel exist in their education. This leads to the following skills being defined as gaps: Docker, Cloud Development Tools, C#, Typescript, and Requirement Scoping. The gaps can also be somewhat identified based on student interaction with the skills with the highest interaction skill being considered the greatest gap to the students. With this in mind a preliminary insight that students may feel that communication and cloud development tools are gaps within their education as these two skills were by far the most interacted with. Additionally, RQ1 and RQ2 mostly cannot be answered by this study as this study did not allow for the industry professionals or the students to propose skills that they view as gaps and insight sought their insight into a predetermined set of skills that may not coincide with an actual gap.

RQ 3 is for the most part answered by the evaluation of the Gaps in Software Engineering Education Platform. While overall grand claims about closing these gaps do not feel appropriate to make, the reported interaction with the system at the least shows more information being provided to students. More specifically, over half of the students recruited for the study interacted with the platform. Overall this study makes progress towards addressing gaps that may exist with software engineering education but all of the results from this study should be taken as introductory and can perhaps be an initial framework for future studies.

Bibliography

[1] Andrew Begel and Beth Simon. Novice software developers, all over again. In *Proceedings of the Fourth International Workshop on Computing Education Research*, ICER '08, page 3–14, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582160. doi: 10.1145/1404520.1404522. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/1404520.1404522.

[2] Eric Brechner. Things they would not teach me of in college: What microsoft developers learn later. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, page 134–136, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581137516. doi: 10.1145/949344.949387. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/949344.949387.

[3] Chris Brown and Chris Parnin. Nudging students toward better software engineering behaviors. In *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*, pages 11–15, 2021. doi: 10.1109/BotSE52550.2021.00010.

[4] Dale Callahan and Bob Pedigo. Educating experienced it professionals by addressing industry's needs. *IEEE Softw.*, 19(5):57–62, sep 2002. ISSN 0740-7459. doi: 10.1109/MS.2002.1032855. URL https://doi-org.ezproxy.lib.vt.edu/10.1109/MS.2002.1032855.

[5] Partha Chakraborty, Rifat Shahriyar, and Anindya Iqbal. Empirical analysis of the growth and challenges of new programming languages. In *2019 IEEE 43rd Annual*

Computer Software and Applications Conference (COMPSAC), volume 1, pages 191–196, 2019. doi: 10.1109/COMPSAC.2019.00034.

[6] Marisa Exter. Comparing educational experiences and on-the-job needs of educational software designers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, page 355–360, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326056. doi: 10.1145/2538862.2538970. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/2538862.2538970.

[7] M. F. Farghally. Student perceptions of the complete online transition of two cs courses in response to the covid-19 pandemic. *2021 ASEE Virtual Annual Conference Content Access, Virtual Conference*. URL https://par.nsf.gov/biblio/10294501.

[8] CC2020 Task Force. *Computing Curricula 2020: Paradigms for Global Computing Education*. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450390590.

[9] Vahid Garousi, Gorkem Giray, Eray Tuzun, Cagatay Catal, and Michael Felderer. Closing the gap between software engineering education and industrial needs. *IEEE Software*, 37(2):68–77, 2020. doi: 10.1109/MS.2018.2880823.

[10] Carlo Ghezzi and Dino Mandrioli. The challenges of software engineering education. pages 115–127, 12 2006. ISBN 978-3-540-68203-5. doi: 10.1007/11949374_8.

[11] David Gibson, Nathaniel Ostashewski, Kim Flintoff, Sheryl Grant, and Erin Knight. Digital badges in education. *Education and Information Technologies*, 20:403–410, 2015.

[12] Wouter Groeneveld, Brett A. Becker, and Joost Vennekens. Soft skills: What do computing program syllabi reveal about non-technical expectations of undergraduate students? In *Proceedings of the 2020 ACM Conference on Innovation and Technology in*

Computer Science Education, ITiCSE '20, page 287–293, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368742. doi: 10.1145/3341525.3387396. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3341525.3387396.

[13] Wouter Groeneveld, Joost Vennekens, and Kris Aerts. Identifying non-technical skill gaps in software engineering education: What experts expect but students don't learn. *ACM Trans. Comput. Educ.*, 22(1), oct 2021. doi: 10.1145/3464431. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3464431.

[14] Dianne Hagan. Employer satisfaction with ict graduates. In *Proceedings of the Sixth Australasian Conference on Computing Education - Volume 30*, ACE '04, page 119–123, AUS, 2004. Australian Computer Society, Inc.

[15] Qiang Hao, Brad Barnes, Robert Maribe Branch, and Ewan Wright. Predicting computer science students' online help-seeking tendencies. *Knowledge Management & E-Learning: An International Journal*, 9(1):19, 2017.

[16] Sara Hooshangi, Ryan Buxton, and Margaret Ellis. 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*, ITiCSE '22, page 61–67, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392013. doi: 10.1145/3502718.3524802. URL https://doi.org/10.1145/3502718.3524802.

[17] Barbara Kitchenham, David Budgen, Pearl Brereton, and Philip Woodall. An investigation of software engineering curricula. *J. Syst. Softw.*, 74(3):325–335, feb 2005. ISSN 0164-1212. doi: 10.1016/j.jss.2004.03.016. URL https://doi-org.ezproxy.lib.vt.edu/10.1016/j.jss.2004.03.016.

[18] Mehmet Kosa, Murat Yilmaz, Rory O'Connor, and Paul Clarke. Software engineering education and games: A systematic literature review. *Journal of Universal Computer Science*, 22:1558–1574, 12 2016.

[19] Sooun Lee and Xiang Fang. Perception gaps about skills requirement for entry-level is professionals between recruiters and students: An exploratory study. *Inf. Resour. Manage. J.*, 21(3):39–63, jul 2008. ISSN 1040-1628. doi: 10.4018/irmj.2008070103. URL https://doi-org.ezproxy.lib.vt.edu/10.4018/irmj.2008070103.

[20] Leo A. Meyerovich and Ariel S. Rabkin. Empirical analysis of programming language adoption. *SIGPLAN Not.*, 48(10):1–18, oct 2013. ISSN 0362-1340. doi: 10.1145/2544173.2509515. URL https://doi.org/10.1145/2544173.2509515.

[21] Ana M. Moreno, Maria-Isabel Sanchez-Segura, Fuensanta Medina-Dominguez, and Laura Carvajal. Balancing software engineering education and industrial needs. *J. Syst. Softw.*, 85(7):1607–1620, jul 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2012.01.060. URL https://doi-org.ezproxy.lib.vt.edu/10.1016/j.jss.2012.01.060.

[22] Laurie Murphy and Josh Tenenberg. Do computer science students know what they know? a calibration study of data structure knowledge. *SIGCSE Bull.*, 37(3):148–152, jun 2005. ISSN 0097-8418. doi: 10.1145/1151954.1067488. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/1151954.1067488.

[23] Pan-Wei Ng and Shihong Huang. Essence: A framework to help bridge the gap between software engineering education and industry needs. In *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 304–308, 2013. doi: 10.1109/CSEET.2013.6595266.

[24] Damla Oguz and Kaya Oguz. Perspectives on the gap between the software industry

and the software engineering education. *IEEE Access*, 7:117527–117543, 2019. doi: 10.1109/ACCESS.2019.2936660.

[25] Sofia Ouhbi and Nuno Pombo. Software engineering education: Challenges and perspectives. In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 202–209, 2020. doi: 10.1109/EDUCON45650.2020.9125353.

[26] Walter Schilling. Wip: Integrating modern development practices into a software engineering curriculum. In *2022 ASEE Annual Conference & Exposition*, Minneapolis, MN, August 2022. ASEE Conferences. https://peer.asee.org/41309.

[27] Nigar M. Shafiq Surameery and Mohammed Y. Shakor. Use chat gpt to solve programming bugs. *International Journal of Information Technology & Computer Engineering (IJITC) ISSN : 2455-5290*, 3(01):17–22, Jan. 2023. doi: 10.55529/ijitc.31.17.22. URL http://journal.hmjournals.com/index.php/IJITC/article/view/1679.

[28] Markus Weinmann, Christoph Schneider, and Jan vom Brocke. Digital nudging. *Business & Information Systems Engineering*, 58:433–436, 2016.

Appendices

Appendix A

First Appendix

A.1 Section one

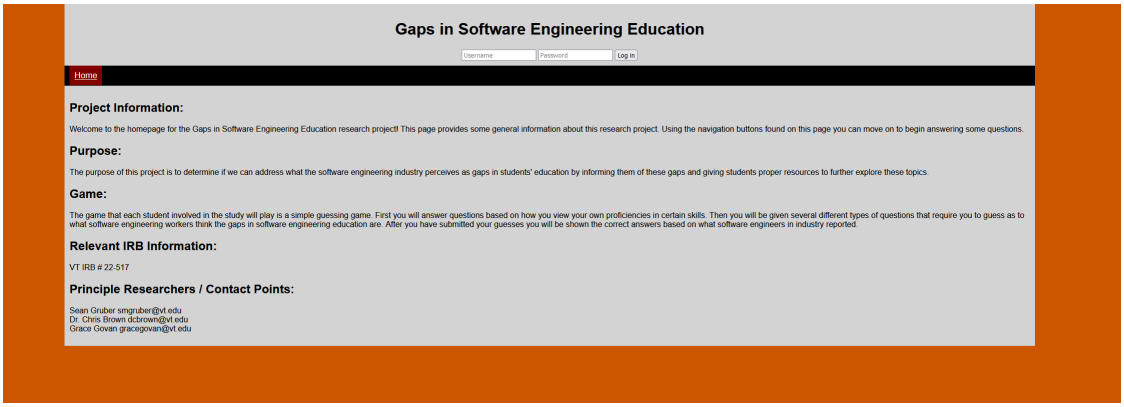


Figure A.1: Home screen of the Platform

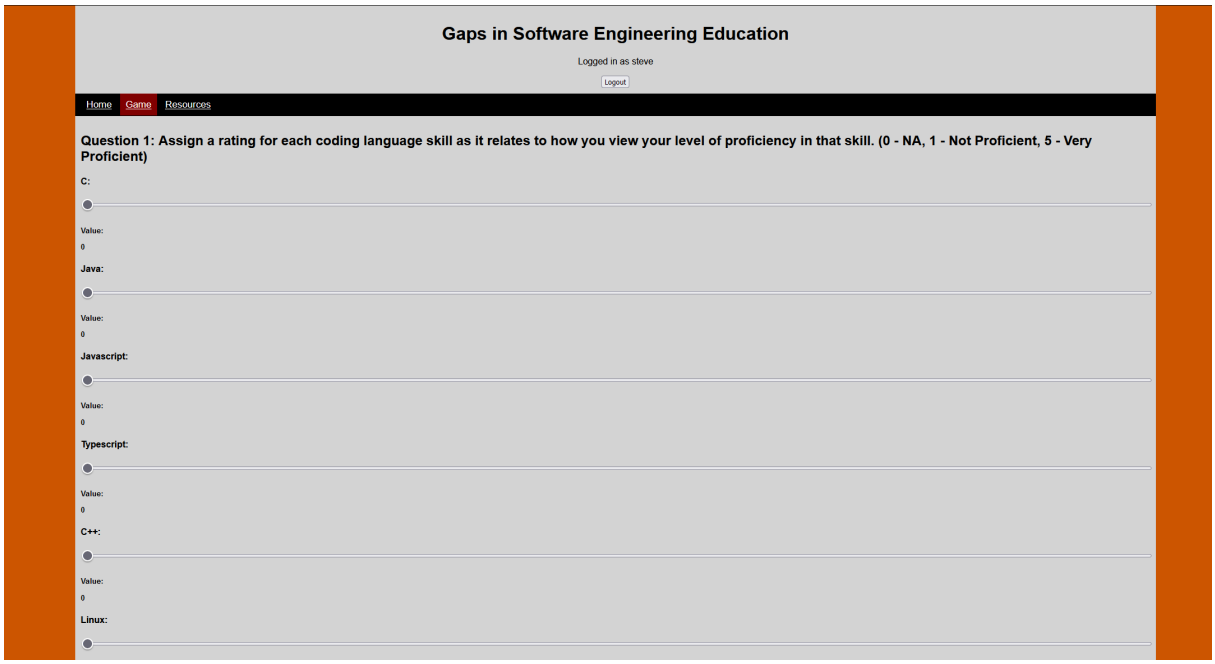


Figure A.2: Likert Scale question for coding skills

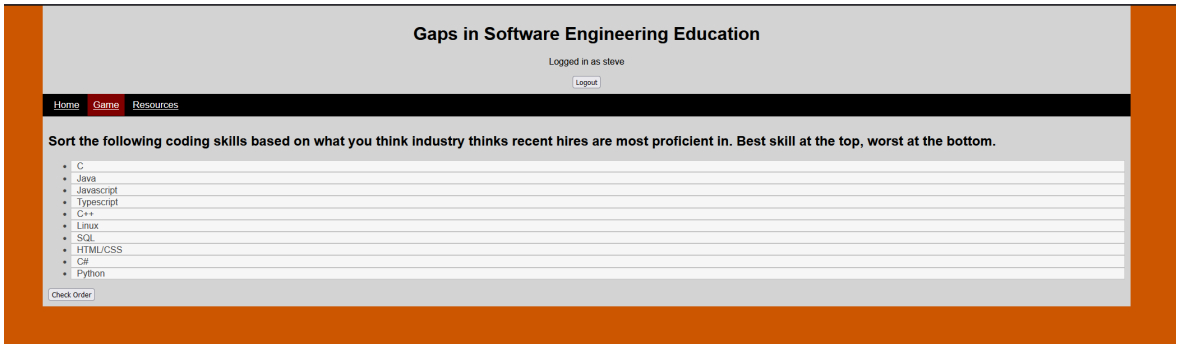


Figure A.3: Guessing game question

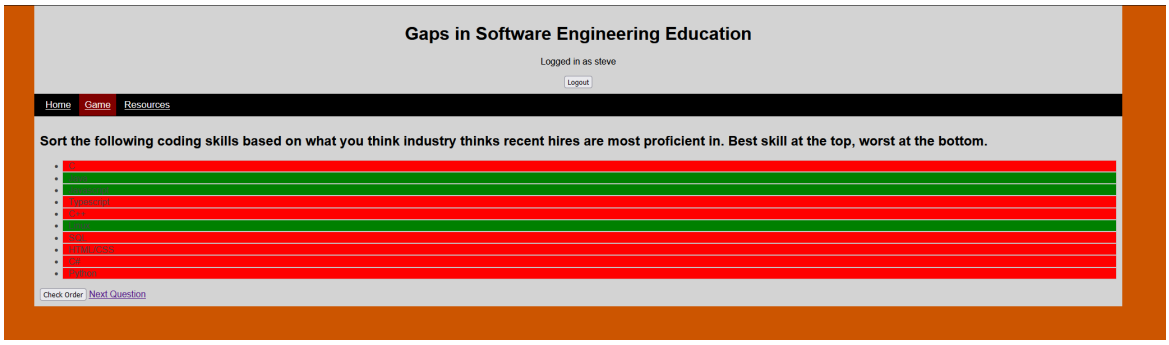


Figure A.4: Guessing game question where a guess has been submitted.

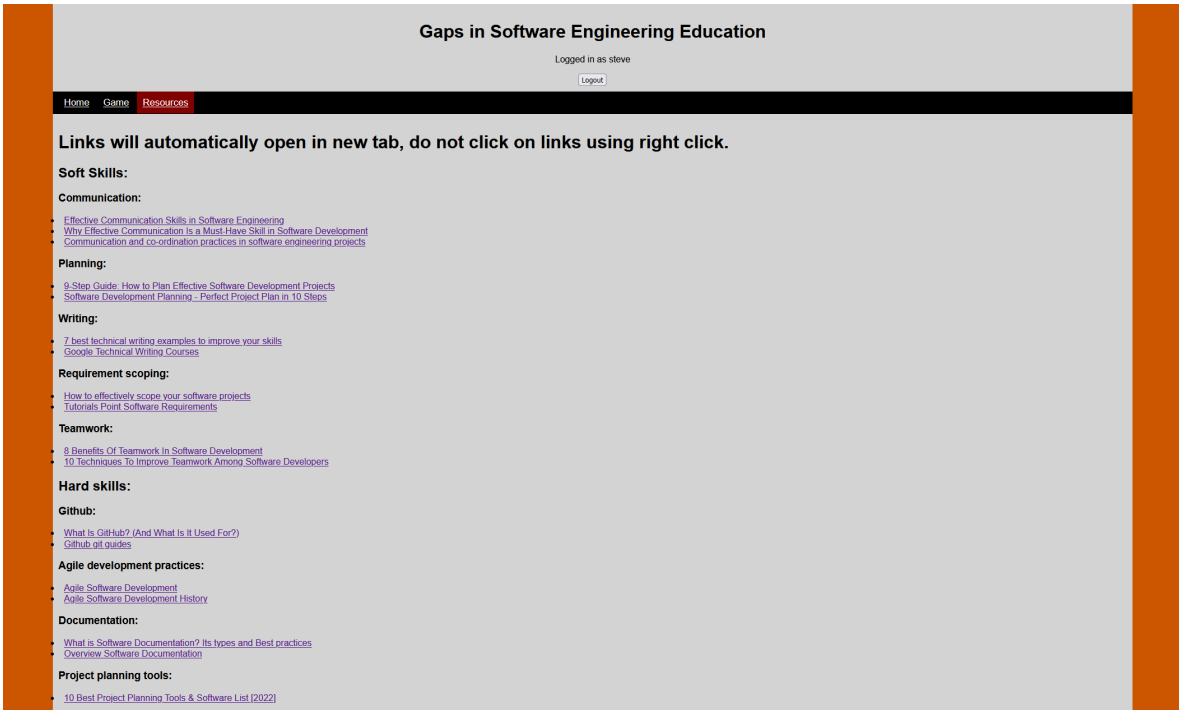


Figure A.5: Resources Page

A.2 Surveys

A.2.1 Industry Survey

Gaps in Software Engineering Education

,

Hello:
You are invited to participate in our survey about gaps in software engineering education. In this survey, you will be asked to answer questions that pertain to your experience working with new hires who are fresh college graduates and you will be asked to evaluate them on several skills. It will take approximately 10 minutes to complete the questionnaire.

Your participation in this study is completely voluntary. There are no foreseeable risks associated with this project. However, if you feel uncomfortable answering any questions, you can withdraw from the survey at any point. It is very important for us to learn your opinions.

Your survey responses will be strictly confidential and data from this research will be reported only in the aggregate. Your information will be coded and will remain confidential. If you have questions at any time about the survey or the procedures, you may contact Dr. Chris Brown (dcbrown@vt.edu).

Thank you very much for your time and support. Please start with the survey now by clicking on the **Continue** button below.

* 1. What company do you work for?

* 2. What position do you hold within your company?

* 3. How many years of software development experience do you have?

* 4. How many recent college graduates have you worked with or mentored in the past 5 years?

- 0
- 1-3
- 3-5
- 6+

* 5. Assign a rating for each soft skill as it relates to the skill of new hires that you have worked with. (0 - NA, 1 - Not proficient, 5 - Very proficient)

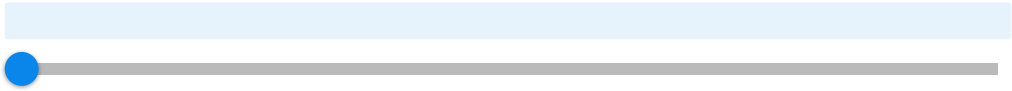
Communication (How well do they communicate with the team?)

Planning (How well do they plan out their tasks?)

Writing (Writing documents like memos, project proposals, etc.)

Requirement scoping (How well they can scope what is required to accomplish their task?)

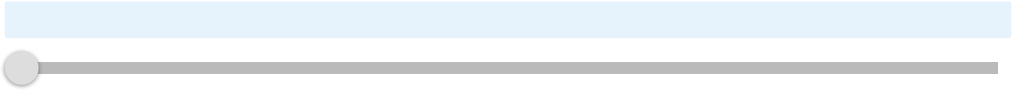
Teamwork (Do they work well with the team? IE Do they try to work with team members or accomplish tasks on their own?)



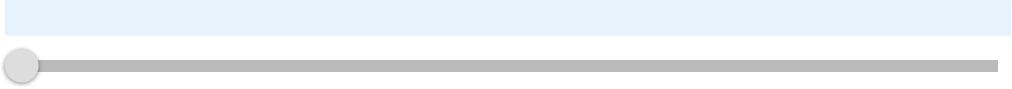
6. Elaborate on any score you gave in the soft skills section. (Not required)

* 7. Assign a rating for each hard skill as it relates to the skill of new hires that you have worked with. (0 - NA, 1 - Not proficient, 5 - Very proficient)

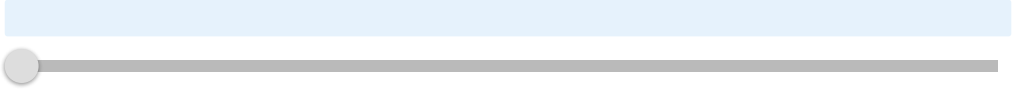
Github (This includes whatever code review process your company uses.)



Agile development practices (Specific practices related to agile. IE Task point assignment, daily standup updates etc.)



Documentation (Includes how well they comment their code and how useful the documents they create are.)



Project planning tool (Trello or similar planning tool)

Docker (or similar build tool)

Cloud Development Tools (AWS or similar)

Unit Testing (How well they test their own code)

8. Elaborate on any score you gave in the hard skills section. (Not required)

* 9. Assign a rating for each coding language skill as it relates to the skill of new hires that you have worked with. (0 - NA, 1 - Not proficient, 5 - Very proficient)

C

Java

Javascript

Typescript

C++

Linux

SQL

HTML/CSS

C#

Python

10. Elaborate on any score you gave in the coding language skills section. (Not required)

* 11. Would you be willing to participate in an interview to elaborate further on your responses?

- Yes
- No

12. Contact Information

First Name

Last Name

Phone

Email Address

A.2.2 Retrospective Survey

,

Gaps In Software Engineering Education Retrospective

This survey is to gather information about the use of the Gaps In Software Engineering Education platform.

* Indicates required question

1. Did you use the Gaps in Software Engineering Education Platform? *

   Mark only one oval.

   - ◯ Yes *Skip to question 3*
   - ◯ No *Skip to question 2*

Gaps In Software Engineering Education Retrospective

2. (Optional) Provide any reasoning for why you did not use the platform.

Gaps In Software Engineering Education Retrospective

3. Did you use any of the resource links provided? *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

4. If you answered Yes to the previous question please give an estimate on the number of resources you used.

5. Did any of the industry gaps surprise you? *

   Mark only one oval.

   - Yes
   - No

6. If you answered yes to the previous question please feel free to provide any comments to any gaps that surprised you.

7. Do you have any suggestions to improve the Gaps In Software Engineering Education Platform?

Google Forms