INFO1111: Computing 1A Professionalism

2022 Semester 1

Practice: Team Project Report

Submission number: 01

Team Members:

Name	Student ID	Levels being attempted in this submission
Xu Wang	**510548788**	**1, 2**
Yiwei Bian	**520430468**	**1**
Yangyang Zhang	**520464485**	**1**
Yi-Yang Wang	**520508321**	**1, 2**

Contents

General Instructions

You should use this LaTeX template to generate your team project report. Keep in mind the following key points:

- When we assess your report, you are not given a mark. Instead we will indicate (separately, for each team member) whether each level is "achieved".
- In order to pass the unit, you must achieve at least level 1.
- In order to achieve level 2, you must first have achieved level 1, and so on for each level up to level 4. This means that we will not assess a higher level until a lower level has been achieved (though we will review one level higher and give you feedback to help you in refining your work).
- Some parts of the report are completed as a team and other parts require each student to complete a different section. This means that for each submission, some members of the team may have completed their work for a given section, but other members may not. It also is therefore possible that some members of the team may achieve a specified level and other members of the team may not yet have achieved that level.
- Even if some members are completing their material for a given level, and others are not, your team members will still need to work together to edit and compile the report. The only exception to this is where a member of the team has already achieved the level they are targeting in a previous submission and has decided to not attempt higher levels, and so is not contributing any further (this should be obvious because no level is indicated for that student on the cover page).
- When completing each section you should remove the explanation text and replace it with your material.

For each submission you will add new details to this report, and/or update previous sections (where previous work was not good enough to have achieved the relevant level). In particular:

- **General:** For each submission, each student can attempt up to 2 levels. You must also successfully achieve each lower level before you can be assessed at a higher level. For example, in the first submission you might attempt only level 1, but not be successful in achieving that level. You then reattempt level 1 and add in level 2 in the second submission and are successful in achieving level 1 but not level 2. For the third and final submission you could then attempt level 2, or levels 2 and 3 - or even just choose to not submit anything further and remain at level 1).
- **Submission 1:** You should complete at least the material for level 1 (since achieving level 1 is required to pass the unit). Each member of the team can also optionally choose to complete the material for level 2.
  Note 1: If you do not complete the level 2 information then you obviously cannot achieve level 2 at this stage. This does not stop you from attempting level 2 in Deliverable 2 or 3, but it will make it more difficult to achieve the higher levels later in the semester. Note 2: To be able to achieve Level 1 in submission one your team has to achieve level 1 in the group component (Section 1.1) and you have to achieve Level 1 in the individual component (i.e. your assigned section 1.2, 1.3, 1.4 or 1.5)

- **Submission 2:** Each member of your team will complete additional sections, but because you are submitting a single document, you need to work together to compile your results together and generate the final submission.
  If you did not achieve level 1 in your first submission, then you should revise the material for level 1 based on the feedback, and optionally you can also complete level 2.
  If you achieved level 1 in your first submission, then each team member can optionally complete the material for levels 2 and 3. *Note: If you do not achieve level 1 with this submission then the highest level you will be able to achieve in the final submission will be level 2. If you achieve level 1, but not level 2, with this submission then the highest level you will be able to achieve with the final submission is level 3.*

- **Submission 3:** Again, you can correct sections where you did not achieve the specified level in the previous submission, and you complete additional sections.
  If you still have not achieved level 1, then you should revise the material for level 1 based on the feedback, and again optionally you can also complete level 2.
  For those at level 1, you can choose to complete the material for levels 2 and 3.
  For those at level 2, you can choose to complete the material for levels 3 and 4.
  For those at level 3, you can choose to complete the material for level 4.

Whilst the team project is just that – a team project – it has been designed to also allow different members of the team to achieve different outcomes. We do expect you to work together as a team. If you do come across problems working together then the first step should be to discuss this with your tutor. Note: If you are having problems you should approach your tutor as soon as you can to make them aware of the difficulties you are having with your team.

Finally, you should also ensure that any resources you use are suitably referenced, and references are included into the reference list at the end of this document. You should use APA 6th reference style (?, ?).

1. Level 1: Basic Skills

Level 1 focuses on basic technical skills (related to LaTeX and Git) and the types of skills used in different computing jobs.

1.1. Developing industry skills

This section is completed as a team.
Throughout your Computing degree we will help you learn a range of new skills. Once you graduate however you will need to continue to learn new languages, new tools, new applications, etc. For this section you need to identify 5 approaches you can take to this continual learning. You should then put these in order from most effective to least effective, and then explain the circumstances in which each approach might be appropriate. (Target = ∼100 words per skill = ∼500 words total).

Demand driven

Individuals working in computer science related field are often pushed by external factors to perpetual learning due the fast integrating technologies, languages applications in the computing field. These external factors faced by an individual inevitably leads to the action of continual learning in the need to solve problems and challenges at work with a more efficient and more suitable method. Without the process of continual learning, the lack of knowledge to the newly integrated computing field could cause career stall or even replaced in the workforce, which is a common phenomenon in IT companies who have superabundant human resources in the younger generations. In addition, the insufficient knowledge of the most current technologies in the industry leads to lost shares for market leaders, which is also an undesirable outcome. To sum up, demand driven continual learning is crucial to individuals in the computing field who wants to have a sustainable and distinguished career, therefore making it the most important approach in the attempt of continual learning.

Study groups: social interactions

Establishing social connections within the learning community can effectively help expand the learner's knowledge base through group work and communication (Laal & Ghodsi, 2012). Learners will have the opportunity to share their individual ideas and knowledge within the group to not only help other learners learn, but also to enhance their own understanding in the topic of study through phrasing and vocalizing their ideas. Through sharing and discussing conflicting thoughts, learners will be able to develop critical thinking skills through approaching problems from multiple perspectives and learning to resolve problems using different methods from ones they are used to.

Feedback

During the process of learning, individuals require effective feedback to act as guidance and motivation in learning (Vollmeyer & Rheinberg, 2005). In both oral and written form, feedback provides individuals with an 'assessment' of their progress, and whether or not it is meeting their individual demand for learning outcomes. Moreover, feedback also allows individuals to gain insight into the effectiveness of their current approach to learning. If this feedback is unsatisfactory, this acts as a form of reflection and incentivises the possibility for further improvements if the current result is satisfactory.

Constructivism

Appropriate when learning individually, and needs to digest the knowledge based on personal construction of meaning by the learner through prior experience. Further, information can be passively received, while not being actually processed by an individual. So by making use of resources such as books, journals, online tutorial videos to develop basic prior knowledge, individuals need to construct meaning of new knowledge internally by understanding the logic behind concepts through using their own belief system. This means that individuals understand information from a distinct point of view from others, thereby learning should engage in learning experience through comparing, questioning, challenging, investigating, accepting and discarding information, and focus on personal development (Cooperstein & Kocevar-Weidinger, 2004).

Conference

The technology world is ever evolving. No matter how knowledgeable, no individual can predict or up-to-date about every technological trend. IT professionals can be well aware of the technologies they use at work, however knowing what we don't know is a key step to continual learning. Attending conferences would give professionals opportunities to know what experts and peers do and learn from them. Technical conferences are also a good place to share technical achievements amongst professionals and a good place to obtain feedback on what may need to be improved or at the very least new directions of learning. Attending conferences would potentially solve one's learning difficulties by putting an individual in touch with other professionals in the same field, who might have faced the same issues. Conferences could be utilized to share opinions on the subject and to absorb valuable information.

1.2. Skills: Xu Wang : Computer Science

This section is completed individually. Each member of the team should independently complete a separate copy of this section.
You should begin by allocating to each team member a different major to focus on (i.e. one of: Computer Science; Data Science; Software Development; Cyber Security). *If you have a fifth member, then your tutor will suggest a fifth topic to cover.* You should then undertake research into the typical practical skills that you believe would be most important to someone who graduates with this major and is then working in industry. You should list the 8 skills that you believe are most important and for each one give a short explanation as to why you feel it is important. (Target = ∼100 words per skill ∼800 words total per student).

Eight practical skills that are most important in computer science

Problem Solving

Problem solving skill is commonly known as one of the most important skills in not just STEM graduates but also other professions and industries. Due to a growing digital industry, problem solving has been listed as one of the 21st Century skills (Voogt & Roblin, 2010), also commonly considered as the core of computer science (CS). Specifically, from the prospective of CS professionals, problem solving skill is placed at the top amongst others (Exter, Caskurlu, & Fernandez, 2018). In addition, by interviewing employers,

Lundberg, Krogstie & Krogstie (2021) found that problem solving is the most desired soft skill as in oppose to hard knowledge base skills.

Self-Learning

It is well known that computer technologies evolves fast, therefore in such fast paced industry, individuals, in order to stay competitive, will have to adapt the mentality of life long learning(LLL) (Borjesson, Pareto, Snis, & Staron, 2007). While employability may require certain hard skills with a knowledge base, in order to be recognized as a fully operational professional, A CS graduate must have the ability to absorb knowledge either through workplace training or by taking other approaches such as project oriented learning with a LLL that follows.

Mathematical Logic

Similar to other science related subjects, computer science has strong connections with mathematical logic. In a sense, computers were developed to help better solving mathematical problems in the first place. Computer circuits utilise propositional calculus to form logic gates which determines the logical relationship of inputs and outputs (Ben-Ari, 2012). Even though some overlapping learning path when comparing to a SD major, CS focuses marginally more on scientific theories behind computer operations which mostly based on mathematical logics, making the ability to understand and apply logic crucial.

Communication skills

CS graduate may have large knowledge base of programming languages, yet transforming the technical terms to human readable contents in a easy to understand form might pose challenge to majority. In a survey done to CS professionals, both verbal and written communication skills are placed in the top ten skills from their prospective (Exter et al., 2018). From employers prospective, communication skill is also one of the top qualities they seek in candidates (Lundberg, Krogstie, & Krogstie, 2021).

Critical Thinking

In contrast to the issues we face in learning CS, real world computing problems are usually in an ill-structured form that requires critical thinking. Rated universally as one of the top three non-technical skills amongst CS professionals, critical thinking is also one of the most sought after skills from the perspective of employers. The ability to identify, categorize and search for the most sensible solution is valuable for any profession especially in a innovated and dynamic environment of computing (Baird & Parayitam, 2019).

Team Problem Solving

As a CS graduate, individual can choose a number of career paths in which most certainly involves a variety of degrees of team work and collaboration. Many of the commonly used libraries and software are forked or inspired, if not directly involved of open source software (OSS) which itself is a proven example of open collaboration (Levine & Prietula, 2014). Thereby, the ability of team work is essential, not only within institutions or companies, but also in a boarder sense of CS community.

The ability to connect different roles

CS graduates tend to have a broader career choice, therefore being able to perform in different roles plays an important part in pursuing a well-established CS career. As tech-all-rounders, CS graduates are desired to have the ability to connect the different skill sets within or across teams that play different functionalities in an organization. Being closer to the fundamental theories behind computer operations, CS graduates are arguably best suited to explain technical terms to stakeholders from a non-technology background.

Concepts across languages

The total amount of computer languages is most commonly believed to be over 700. Even though one can never muster the entirety of computing languages, understanding the concepts will help CS graduate to gain advantage while working on specific projects that require several languages to optimize functionality or performance. For example, scripting languages are efficient with small amount of data structures however one may choose an object oriented programming language such as Java or C to build a performance prioritised project. Thereby, it is important for CS graduates to have knowledge across languages and apply the most appropriate.

1.3. Skills: add student 2 name here : Data Science

maths and statistics: mathematical

1.4. Skills: Yangyang Zhang : Software Development

Professionalism

The current society expects individuals to uphold a certain set of moral and ethical stance towards working methods (Myers, Hall, & Pitt, 2012). As such, in order to create a sustainable professional image of software developers (SD)- technical competence, professional responsibility for the software they develop and to themselves and the general society are important skills expected to incorporate. Specifically, Myers, Hall and Pitt (1997) suggests that SD are expected to uphold the reputation of the profession and continuously seek to improve professional standards through engaging in their development, use and enforcement, as well as to avoid actions that affect the good standing of the profession. Similarly, SD should also not misrepresent or conceal information on the capabilities of products, systems or services with which they are concerned or unethically take advantage of the lack of knowledge and experience of others.

Expertise in software development methodologies

Software development methodologie acts as a platform for developers to cooperate together and communicate with greater efficiency. For example, despite traditional and agile methodologies, the popularity of hybrid methodologies are now increasing. To elaborate, teams need to choose and use different methodologies in projects based on contextual measures (Vijayasarathy & Butler, 2015). For example, the prominent agile development methodology includes Crystal and Extreme programming- have the advantages of allowing software to be released in iteration, thereby improving efficiency. While compared to DevOps deployment methodology, incorporates the benefits of improving time to market, lowering the failure rate of new releases and shortening the lead time between fixs (Synopsys, 2017). Thereby, this requires SD graduates to possess skills in determining the suitability of specific methodology. For example, agile practices are more useful in small

to medium teams working on less complex projects. Inevitably, SD graduates need to gain insightful understanding in different forms of methodologies to develop their own hybrid SD methodology rather than strictly following practices of a predefined methodology (Williams, 2010).

Expertise in programming language

As a SD graduate, Lee, shjhd (2019) discovered that integral to further development in the workplace, higher levels of technical skills such as knowledge in programming languages such as Java, python and Linux in early stages are very important. To elaborate, knowledge acts as the basis to developing software, and as a graduate, expertise in at least 2 programming languages is very important since each programming language has its advantages and disadvantages. Meaning SD graduates need to be familiar with at least 2 languages to maintain competitiveness and ensure efficiency in the workspace. For example, Java provides automatic garbage collection and is relatively easy to use, however provides no backup facility and requires significant memory space. While comparatively, C++ requires low level manipulation, has memory management and offers the features of portability. Despite having the disadvantages of requiring the use of pointers and the absence of garbage collectors (Jain, Sonar, Jain, Daga, & Jain, 2020) Thus based on different context, SD graduates need to choose the most suitable programming language, and thus requires expertise in at least 2 programming languages.

interpersonal skills

Interpersonal skills including communication, problem solving and critical thinking is very important for SD graduates as developers usually work in a team. Specifically, effective communication between team members improves efficiency of progress, and allows developers to understand the authentic needs of clients. Further, The software development industry evolves and changes quickly, the diversity of new techniques and approaches, and different clients and context requires critical thinking- the ability to judge correctly and to discriminate successfully among different issues to maintain competitiveness (Chouseinoglou & Bilgen, 2014). This idea is integral to the skill of problem solving, which is also essential as graduates need to be able to discover the core of the issue and develop effective strategies to resolve it. For example, graduates need to be able to debug their program and resolve conflict within team members or with clients.

Project management skills

Software development projects tend to be complex and multi-faceted. Thereby making project management skills including planning, organization and monitoring process extremely important in ensuring efficiency (Orases, n.d.). Planning is an important skill in teamwork, as team members without a clear vision or guidance often encounter errors and confusion, thus extensive planning before project execution would ensure efficiency in SD. This is integral to the skill of oganising, where graduates may be required to manage and set deadlines for team members to maintain the progress of projects. Further, during the SD process, unexpected external risks and changes require managers to establish and execute a defined project quality standard to ensure the project delivered meets customer expectations despite possible changes that may occur. Hence it is important to monitor the final product and establish corrective procedure if required.

Data structure and Algorithms

When software developers use programming languages, data structure and algorithms are also employed internally. As data structure is a method for organising and storing data in computers, it represents the relationships between those values and the functions and operations they perform. There are two types of data structure: linear data structure (Arrays, Stack, Queue) and non-linear data (Tree, Graph, map) structure. The two types possess major differences, where most notably, the data items in linear data structure are arranged in sequential order and all items are present on the single layer, where non-linear data items are arranged in a hierarchical manner and data items are present at different layers (Programiz, n.d.). This is supported by the use of algorithms, which are structured computational procedures that transform input values to an output. Where effective use of algorithms will ensure efficiency of computer programs and proper utilization of resources (Mulongo, 2022). This means that graduates need to obtain knowledge about data structure as it is an effective system to retrieve and manage large datasets and is essential for graduates to understand and choose the most appropriate data structure for projects.

selflearning

The software industry is developing rapidly, hence requiring graduates to be flexible and adaptable to changes and respond to external influences through active self learning. This continuously evolving industry is constantly filled with new innovation and ideas that drive changes in the industry- such as trending frameworks and new developer tools . Thereby requires graduates to be proactive in resolving insufficient gaps of knowledge, and digest unfamiliar information in order to fulfill expectations and maintain competitiveness. This skill requires the graduate to have the ability to identify missing knowledge, gather relevant information and understand its logical reason by constructing meaning that is specifically developed through personal thinking.

Teamwork and collaboration

Software development often requires graduates to work in a team, thereby facilitating the skill of collaborating with others. Particularly, contribution in team discussion shares experience and knowledge between members, and enhances the efficiency of the project when it is employed in conjunction with the skill of communication. Moreover, active participation allows team members to receive feedback on their work through peer evaluation, overall facilitating a better result. Further, using collaborative development flatform such as GitHub also enables better efficiency of projects, as individuals can work on their own parts on their own device and on the part they are confident or specialise in, and combine the final result anytime they like.

1.5. Skills: Yiwei Bian : Cyber Security

Open mindedness / Adaptability

The field of computer and technology is developing rapidly, and threats to systems are not guaranteed to come from old technology or methods that you are familiar with. One should always be open and informed about modern technologies and methods that may not necessarily be in frequent use but are growing rapidly so that one would not be bewildered when unfamiliar sources of threats and attacks appear. Even if a system is subject to unknown threats and attacks, it is important to be able to keep motivated to

learn new knowledge in unfamiliar topics, using and learning the new knowledge efficiently to produce defence methods.

Problem solving / critical thinking

It is important to look and think from multiple perspectives on a problem or a system to look for vulnerabilities or flaws that could be subject to attacks. Different forms of attacks and threats may come from so many sources and in so many different forms, from troubleshooting in the hardware level, to bugs in the software, even to the potential problems arising from people that is accessing the system. Thus, it is important that one's thoughts are not stuck down a single path, but able to think volatilely according to the problem, and to think openly for all reasons and sources of the threats or attacks rather than those that one is used to dealing with.

Teamwork & communication

Rarely is a single person in charge of a whole project or system in cybersecurity, where more often, teams are employed, making the skills of teamwork and communication an indispensable skill (Liu, Jajodia, & Wang, 2017). Working collaboratively is effective and efficient in both progressing on the project and enhancing one's knowledge around the area. Effective communication and teamwork allow a complex problem to be tackled in smaller pieces efficiently, with team members able to contribute individual thoughts which enriches the collective solution. While conveying disagreements enables each member in the team to be able to approach the issue with a new perspective and potential solution.

Research

To progress along with the rapidly developing field of cybersecurity, research skills are necessary in looking for information about modern methods of attacks and vulnerabilities or keeping up to date with changes and updates in systems. Knowing means to access desired information allows an efficient process of study. Using a wide range of sources, from the internet, videos, to books, human resources, allows one to be able to gain useful information effectively. Not only does research skills help resolving urgent issues which had come up unexpectedly, but continuous research over time also equips one with a solid and wide knowledge base that will serve one well during one's time working in the field.

Self-motivation

To be able to continuously learn and word in the industry, it is important to be able to motivate oneself after extended periods of learning. Positive attitude and mindset are required for one to be able to persist doing something, whether learning or working, continuously for an extended period. Looking for bugs or vulnerabilities in millions of lines of code requires a great amount of effort and patience to do so, and it is important that one can motivate oneself during the tedious process to be able to work efficiently.

Ethics

Working in the field of cybersecurity will indefinitely give rise to many ethical issues and will be exposing one continuously to ethical choices. Whether it is work ethics on one's attitude towards others and their task, or ethics on legal issues. One must realise that the nature of their work is associated with responsibility in ensuring the safety of

certain system, project, or information, and one needs to make ethical consideration about the situation. Other than legal consequences, unethical decisions may expose others and oneself to dangerous situations (Weber & Studer, 2016). Thus, it is important for one to be able to make ethical decisions while working in the field.

Cybersecurity basics

It is necessary for one to have basic knowledge about networking and the system or project that one is working with or around. One needs to know how to manipulate software and tools to preform basic operations such as monitoring package transfer through network connections. It is also important that one is familiar with the system or structure that they are working with, knowing the potential vulnerabilities that may be subject to attacks, and solutions in dealing with these threats, so that one can react efficiently to unexpected and urgent situations.

Diverse background knowledge in the IT field

Other than knowing about cyber security basis and the projects or tools you are required to use; experience in other fields not limiting to computer science may also contribute to working around cybersecurity (Dawson & Thomson, 2018). Through gaining knowledge in fields other than the one you are working in; you may be able draw connections between the two areas and develop further understandings in both or recognize complex issues which occurs in multiple layers of system. For example, programming skills may not be required to work in a cyber security related environment but knowing programming basics may help one understand the most basic layer of their project or system and thus able to work with programmers and recognize issues at another level.

2. Level 2: Basic Technology

Level 2 focuses on initial evaluation of the tech stack that is used by a selected company. All companies make use of a range of technologies, and these technologies need to work together. A tech stack is basically just this collection of technologies that collectively enable a company's systems. As an example, one of the most common technology stacks for supporting web servers is LAMP: Linux as the underlying operating system; Apache as a web server; MySQL as the supporting database; and Perl (or more recently PHP or Python) as the programming language.

Each student should choose a different tech stack and explain the role of each of the different technologies in that stack. Note that prior to researching your proposed tech stack and spending time writing about it, it might be a good idea to check with your tutor as to whether your chosen stack is suitable. (Target = ∼200-400 words per student).

2.1. Tech Stack: Shopify, Ruby on Rails

Shopify is a multi-channel e-commerce platform for small and medium businesses, it requires low-code for businesses to open a online shop and there are thousands of applications on the Shopify platform to support the shops.

Rails is a web application development framework written in the Ruby programming language. Shopify tech stack is famously known to be one of the companies that use Ruby on Rails framework at the core. Shopify is also known to be one of the oldest and biggest Rails application.

Within the Rails stack, they use MySQL as a relational database; memcached for key/value storage; and Redis for queues and background jobs, which remains fundamentally unchanged over the years.

In Shopify's current Rails stack, they use the concept of 'pods' to handle hundreds of applications on Rails. One pod is an independent instance of Shopify containing datastores like MySql, memcached and Redis, so that service interruption of one pod will not have global effects.

To manage all the 'pods', shopify uses Docker(Docker is a set of products that use OS-level virtualization to deliver software in packages), Kubernetes(open-source system for automating deployment, scaling, and management of containerized applications), and Google Kubernetes Engine(A service to automatically deploy, scale, and manage Kubernetes) to make it easy to bootstrap resources for new Shopify Pods.

On the web server level, Shopify uses Nginx, with OpenResty in which the web developers can use the Lua programming language to script various existing nginx C modules and Lua modules and construct high performance web applications and scriptable load balancers (Shatrov, 2018).

2.2. Tech Stack: add student 2 name here

Your text goes here

2.3. Tech Stack: add student 3 name here

Your text goes here

2.4. Tech Stack: add student 4 name here

Your text goes here

3. Level 3: Advanced Skills

Level 3 focuses on more advanced technical skills (LaTeX and Git) and analysis of linkages and relationships between the items in the company tech stack.

The following is a list of advanced Git and LaTeX skills/features. Each student should select one pair of items from each list and demonstrate actual use of each item (either through activity in Git, or through including items in this report). (Target = ∼100 words per student for each feature).

- Git
  - Rebasing and Ignoring files
  - Forking and Special files
  - Resetting and Tags
  - Reverting and Automated merges
  - Hooks and Tags
- LaTeX
  - Cross-referencing and Custom commands
  - Footnotes/margin notes and creating new environments
  - Floating figures and editing style sheets
  - Graphics and advanced mathematical equations
  - Macros and hyperlinks

3.1. Advanced features: Xu Wang

Explain your use of the advanced Git and LaTeX features.

3.2. Advanced features: add student 2 name here

Explain your use of the advanced Git and LaTeX features.

3.3. Advanced features: add student 3 name here

Explain your use of the advanced Git and LaTeX features.

3.4. Advanced features: add student 4 name here

Explain your use of the advanced Git and LaTeX features.

4. Level 4: Advanced Knowledge

Level 4 focuses on analysing your particular tech stack and considering alternatives. Each student should consider the tech stack they described for Level 2, and then discuss each of the following points:

- What are the strengths and limitations of this stack? (Target = ∼200 words).
- What alternatives exist, and under what situations might these alternatives be a better choice? (Target = ∼200 words).

4.1. Advanced Knowledge: Xu Wang

Your text goes here

4.2. Advanced Knowledge: add student 2 name here

Your text goes here

4.3. Advanced Knowledge: add student 3 name here

Your text goes here

4.4. Advanced Knowledge: add student 4 name here

Your text goes here

References

Baird, A. M., & Parayitam, S. (2019). Employers' ratings of importance of skills and competencies college graduates need to get hired: Evidence from the new england region of usa. *Education+ Training*.

Ben-Ari, M. (2012). *Mathematical logic for computer science*. Springer Science & Business Media.

Borjesson, A., Pareto, L., Snis, U. L., & Staron, M. (2007). Continuing professional development by practitioner integrated learning. In *Companion to the 22nd acm sigplan conference on object-oriented programming systems and applications companion* (pp. 897–907).

Chouseinoglou, O., & Bilgen, S. (2014). Introducing critical thinking to software engineering education. In *Software engineering research, management and applications* (pp. 183–195). Springer.

Cooperstein, S. E., & Kocevar-Weidinger, E. (2004). Beyond active learning: A constructivist approach to learning. *Reference services review*.

Dawson, J., & Thomson, R. (2018). The future cybersecurity workforce: going beyond technical skills for successful cyber performance. *Frontiers in psychology*, *9*, 744.

Exter, M., Caskurlu, S., & Fernandez, T. (2018). Comparing computing professionals' perceptions of importance of skills and knowledge on the job and coverage in undergraduate experiences. *ACM Transactions on Computing Education (TOCE)*, *18*(4), 1–29.

Jain, S. B., Sonar, S. G., Jain, S. S., Daga, P., & Jain, R. S. (2020). Review on comparison of different programming language by observing it's advantages and disadvantages. *Research Journal of Engineering and Technology*, *11*(3), 133–137.

Laal, M., & Ghodsi, S. M. (2012). Benefits of collaborative learning. *Procedia-social and behavioral sciences*, *31*, 486–490.

Levine, S. S., & Prietula, M. J. (2014). Open collaboration for innovation: Principles and performance. *Organization Science*, *25*(5), 1414–1433.

Liu, P., Jajodia, S., & Wang, C. (2017). *Theory and models for cyber situation awareness*. Springer.

Lundberg, G. M., Krogstie, B., & Krogstie, J. (2021). From employable to fully operational: The need for training of computer science graduates. *International Journal of Engineering Pedagogy*, *11*(3).

Mulongo. (2022). *The importance of algorithms in computer programming. technotification.* Retrieved 2017-01-31, from `https://www.technotification.com/2019/02/importance-of-algorithms-programming.html`

Myers, C., Hall, T., & Pitt, D. (2012). *The responsible software engineer: Selected readings in it professionalism*. Springer Science & Business Media.

Orases. (n.d.). *The importance of project management for software development.* Retrieved from `https://orases.com/the-importance-of-project-management-for`

-software-development/

Programiz. (n.d.). *Data structure and types. what are data structures?* Retrieved from https://www.programiz.com/dsa/data-structure-types

Shatrov, K. (2018). *E-commerce at scale: Inside shopify's tech stack - stackshare.io.* Retrieved 2018-08-08, from https://shopify.engineering/e-commerce-at-scale-inside-shopifys-tech-stack

Synopsys. (2017). *Top 4 software development methodologies. how do the top software development methodologies (waterfall, rapid application, agile, and devops) work? and which method is best for your project?* Retrieved 2017-03-28, from https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/

Vijayasarathy, L. R., & Butler, C. W. (2015). Choice of software development methodologies: Do organizational, project, and team characteristics matter? *IEEE software*, *33*(5), 86–94.

Vollmeyer, R., & Rheinberg, F. (2005). A surprising effect of feedback on learning. *Learning and instruction*, *15*(6), 589–602.

Voogt, J., & Roblin, N. P. (2010). 21st century skills. *Discussienota. Zoetermeer: The Netherlands: Kennisnet*, *23*(03), 2000.

Weber, R. H., & Studer, E. (2016). Cybersecurity in the internet of things: Legal aspects. *Computer Law & Security Review*, *32*(5), 715–728.

Williams, L. (2010). Agile software development methodologies and practices. In *Advances in computers* (Vol. 80, pp. 1–44). Elsevier.