

# SOFTWARE ENGINEERING HIRING GUIDE

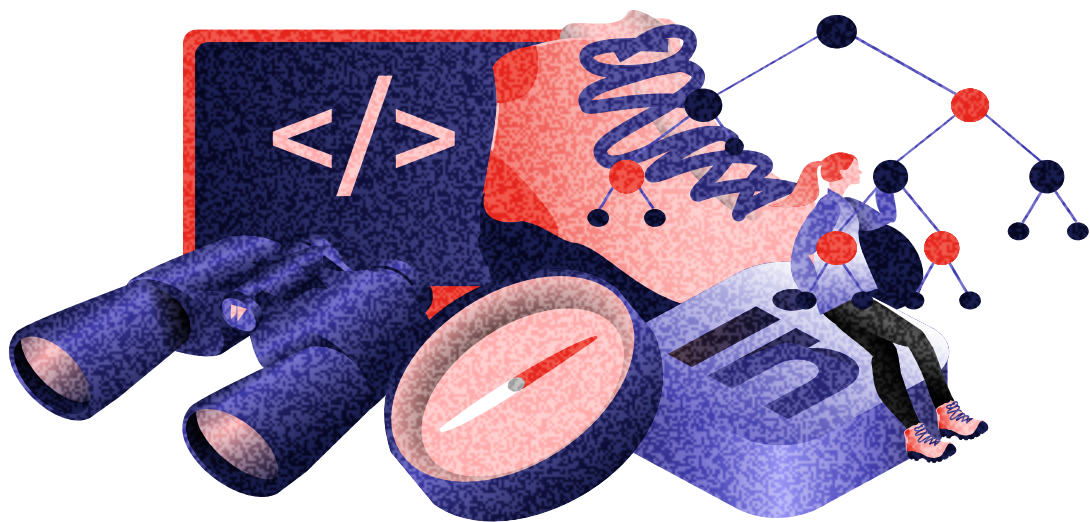
Your guide to navigating the world of tech internships.



QWIC  
COMMUNITY



QUEEN'S  
WOMEN IN  
COMPUTING



# TABLE OF CONTENTS

Landing a tech internship can be a big undertaking and feel like a full-time job in itself. There are many nuances and ways to navigate the industry, so we hope to share more in this guide than just the standard interview tips that you've likely already heard.

2-3 Application Process

4-9 Interview Prep

10 The Actual Interview

11 After the Interview

# APPLICATION PROCESS

Okay there's a lot going on here as this is a tedious stage of the hiring process. If you weren't already sick of the low communication standards amongst school group projects, cue the poor communication with recruiters- where getting 'left on read' may not phase you in the same ways as prior to going through this process. BUT, there's a couple of steps to complete before you can be honored to be 'left on read' by a recruiter :').

First, before you can even start applying to positions, you need a bomb resume. You can refer to our resume guide for the nitty gritty, but here's a high level synopsis of what you should do:

## Single Page

SWE resumes should be strictly technical and focus on measurable impacts.

\*Side Note\* Although soft-skills are extremely important, they aren't often checked until later in the process, if at all. This can also be dependent on the internship you're applying to.

Include your graduation year and links to your LinkedIn, personal website, Github, personal projects, and relevant experiences.

In your skills section, include things like: programming languages, operating systems and frameworks you're familiar with.

Once, you're comfortable with your resume, we HIGHLY HIGHLY recommend you get either personal industry connections or friends with experience applying to SWE jobs to review and edit your resume. Don't take any feedback personally, every bit helps.

Next, we recommend applying to AS MANY (approximately 30 - 100) companies as possible as internships can be extremely competitive, especially if you're applying in the states or if you don't have a lot of prior experience. Now that may sound daunting and redundant. You'll find yourself filling out similar forms that aren't quite identical enough for ctrl + c to save you. However, the reality is that there's a chance you'll never hear back from a certain percentage of these places and another percentage may just send you immediate rejection emails. It sounds harsh, but it's normal to get tons of rejection emails, especially if you're new to recruiting. In the moment it's hard to not get discouraged, but remember it's part of the process and will likely improve as you gain more experience.

Another thing we've learned is that sometimes recruiters are looking for specific skills or qualities not necessarily apparent in the job description. Pinpointing what you could improve in the recruiting process is tricky, because you often don't get feedback or reasoning for why decisions are made. Another benefit of applying to more places is you can use interviews with companies you're less interested in as practice for interviews with companies that you are interested in.

# APPLICATION PROCESS

Once you've submitted your online applications, you enter a crucial step: following-up. If you aren't comfortable putting your trust in a resume sorting algorithm and want to ensure a recruiter sees your resume, you're going to need to do some form of networking. We only recommend following up with the companies that you're most interested in or ones where you feel you might be a more competitive applicant.

LinkedIn is probably one of the most useful places to network. The ability to reach out to strangers on LinkedIn is a strong skill to have in your arsenal. Now you may be thinking, who do I contact and what do I say?

Find someone that works at the company who currently works in the position you're applying for (in this case SWE). Within this subset of people, try to find someone you share similarities with so they're more likely to respond to your invite. This could either be based on a mutual connection, they're alumni of the university you attend, or you share similar experiences and areas of interest.

When you reach out to them, you only have 200 characters to make an impression. Be concise. Explain why you're reaching out and invite them to a coffee chat. The goal of this coffee chat is to establish a genuine connection with this person and learn more about the company, the job, their experiences. By the end they may offer to make a referral on your behalf. If the topic of referrals doesn't come up, you can ask if they could put you in contact with someone related to hiring that you could reach out to.

We also DO NOT recommend reaching out to every SWE at the company. This will come across as ingenuine and hurt you more than help you in the end.

(\*note we wouldn't focus on reaching out to recruiters as they're less likely to respond. Their inboxes are bombarded by people and it's much harder to stand out).

The above tip touched on something called referrals. Referrals are when someone who works at a company recommends an applicant for a job. Companies often have a separate process for referred applicants. It's typically faster and ensures a recruiter sees your application. Some companies are even required to provide a response to the applicant within a certain time period, which can provide the applicant with timely closure or an interview. So when you can, always leverage your network to get a referral.

Now is a good time to reiterate that referrals aren't necessary to get interviews and sometimes aren't always possible. They're just another tool that you can leverage.



# APPLICATION PROCESS

So what happens if you do hear back from a recruiter? First off, congratulations. It's nice to finally feel a bit of acknowledgement for all your sweat from filling out your name 100 times.

The first stage of SWE interviews vary depending on the company, however most commonly it's either an online take-home coding test or an HR interview. Here's a list of common interview evaluation stages:

Stage 1: Online screening: Take-home coding test or HR interview.

Stage 2: Technical online coding interview with an engineer.

(\*very common to have two of these back-to-back with two different engineers)

Stage 3: Final Interview with an engineering manager.

Typically there are 2 - 4 rounds interview rounds. Depending on the company, they may be more technical or behavioural. Typically for internships, they're overwhelmingly technical and minimally behavioural. However, that often depends on the company and internship that you're applying for (ex. Microsoft's Garage Internship is very technical and behavioural). We'll dive deeper into this in the "Interview Prep" section.

Earlier we touched on how it's expected for recruiters to be poor communicators. But they are your point of contact throughout the entire hiring process and between interview rounds, so if you can establish a relationship with your recruiter, it can benefit you.

For example:

1. Recruiters can advocate for you in multiple ways, whether that be getting you an interview, or pairing you with an interviewer better matched to you. At the end of the day they want you to succeed- it's mutually beneficial.

2. There's a higher chance they'll remember you which might make things progress in a more timely manner.

At the end of the day, it's important to advocate for yourself throughout the entire process. If time is starting to pass and your recruiter forgot to contact you, make sure you reach out. Throughout the entire interview process you always want to be PROACTIVE!!

# INTERVIEW PREP

## Interview Preparation

Congratulations on making it to the interview phase of the hiring process! Now it's time to start preparing for your interview.

## Scheduling an Interview

If you're given the option to choose when you interview, try to be strategic about the date you select. While it can be tempting to give yourself as much time as possible to prep by scheduling your interview for the latest possible date, it's important to note that some companies use "rolling applications". This essentially means that it's "first come, first serve" and you have a higher chance of receiving an offer if you interview earlier in the hiring process. If you don't know whether or not your potential company uses rolling admissions, contact your recruiter. Try to avoid interviewing in the middle of midterm season and pushing your interviews back so late that it takes over the next four months of your life.

Sometimes you're not given an option to choose when you interview. If you end up with less prep time, remember other candidates are probably in the same boat as well. Like timing in life, there will never be a "perfect time" and sometimes you just have to go with the flow. If the date and time really doesn't work for you, reach out to your recruiter and they should help you out.

## Preparing for an Interview

Studying for an interview can be an overwhelming undertaking, but it gets easier the more you do it. You should develop a solid plan to organize your studying that follows a "Work smarter, not harder" mentality. Before making an official plan, understand the structure of your interview. Make sure you know the answers to the following questions before starting as knowing these details can help alleviate the stress of an upcoming interview and enable strategic preparation.

Is the interview in person, over video call, or over the phone?

What type of interview is this?

Does it test your technical abilities, behavioural skills, knowledge of system design, and/or understanding of general computer science topics?

Don't waste your time studying concepts that you won't be tested on.

**If it's a technical interview:**

Is it on a whiteboard, an online IDE, or being conducted verbally?

What programming languages can you use?

# INTERVIEW PREP

We encourage you to reach out to friends, friends of friends, or friends of friends of friends (basically anyone you know) who have interviewed at the company before. They can provide some key insights on the structure of the interview and how to best prepare. This has been invaluable to us in the past and we've even made some awesome new friends along the way.

We also highly recommend you read online about the interview process at your potential company. The comments on Glassdoor and Reddit can be lifesavers. You're likely able to find interview questions that companies have asked in the past, and it's not uncommon for them to re-ask them. In your research you may also realize some companies focus on asking different types of questions, so try to account for this in your studying. For example companies like Asana, Palantir, and Facebook have been known to ask harder questions such as recursive 2D matrix questions, whereas most companies generally focus on array and string manipulation questions.

Now that you're almost ready to deep dive into all the material, pick a time of day to regularly devote to interview prep. Consistency is key when preparing for interviews so making it a daily habit can be extremely beneficial.

## **Outline of a potential study plan for a technical interview:**

Start with a refresher on data structures and algorithms. This is meant to be a high level reminder to help jumpstart your brain. The book "Cracking the Coding Interview", YouTube, the website "Geeks for Geeks", and course notes are our favorite resources for doing this.

Practice problems!!!!

Mock interviews and revisiting the topics you find extra challenging.

# INTERVIEW PREP

## Key technical topics to review

\*By no means a comprehensive list\*

### Array and String problems

Familiarize yourself with merge sort, binary search, and the “sliding window” algorithm

### Tree and Graph problems

Understand and be comfortable implementing breadth first search and depth first search.

Know the difference between a binary tree and a binary search tree (they’re easily confused!!)

Practice 2D array questions. These questions typically test recursive backtracking.

### Hash Table problems

Notice their ability to improve the time complexity of your solution since hashing results in  $O(1)$  lookups.

### Recursion problems

Practice, practice, practice!! Recursion can be tricky.

Note that most iterative solutions can also be implemented recursively. Sometimes the recursive solutions are much cleaner.

### Big O Notation

Understand time and space complexity. It’s common to be asked about them and can help you develop better solutions.

These questions can also test your understanding of the coding language you have chosen. For example, Python has many built-in functions you can leverage. If you were using C++, you would have to implement these built-in functions yourself. Hence it’s not uncommon to be asked the time complexities of a particular Python built-in function you’re choosing to use.

### Object Oriented Programming Concepts

Brush up on classes, objects, method overloading, method overriding, encapsulation, inheritance, polymorphism, abstraction, etc.

# INTERVIEW PREP

## Doing practice problems

If you're completing a technical interview, you will most likely be asked to solve a problem for an engineer. We've found that doing practice problems is the best way to prepare for an interview. Our go-to source of practice problems is Leetcode, but there are many other resources out there including Cracking the Coding Interview, HackerRank, and CodeChef.

For those of you who aren't familiar with Leetcode, Leetcode is a free (premium subscription also available) website filled with coding practice questions. The coding questions are sorted into different categories. For example, you can find easy, medium, or hard questions, the top 100 questions to practice if you're interviewing at Google or questions that test your knowledge of binary search trees. You can code your solution in their online IDE and submit your answers for evaluation. You can see other people's accepted answers and read about the various solutions in the comments section. Some engineers who interview candidates source their interview questions directly from Leetcode.

Solving practice problems effectively is an art. You don't want to do 1 million problems but not actually take the time to understand the solutions. You also don't want to spend two weeks completely understanding each of the five possible solutions to a single question. We find the best approach is something in between the two.

# INTERVIEW PREP

Here's a list of tips to keep in mind when solving practice problems:

Set a time limit for each question. It will prevent you from getting too stuck on a problem and help you prepare for your interview where you won't have unlimited time to solve a question.

Make sure you read the full question before brainstorming a solution. You don't want to solve a question that's different from the one being asked.

Start to think of clarifying questions you could ask if you were to get this question in an interview. Sometimes interviewers will test you by not providing "all the information" to the question to make sure you don't just jump into the question. Hence, it's also important to state any assumptions you're making.

Draw a picture. Visualization can be super effective (which can translate to being an effective skill when communicating with your interviewer).

Think about edge cases (arrays with no elements, upper and lower case letters, punctuation in strings, etc.)

Think about multiple solutions. Interviewers appreciate it when you're able to identify different solutions that vary in optimization: ie. stating a brute force solution exists but that optimization x, y, z can be applied. This is also useful if you're initially stuck. You can start with the brute force solution and optimize from there. If you can't solve the optimized one from the start, or even if you can, still state the brute force solution and then solve the optimized one.

Write pseudo code first.

Try the question yourself before looking at the solution. This is important because you won't be able to check the solution in an interview.

If you're really stuck, look for a hint. Even interviewers give hints sometimes.

If you think you've found the solution, walk through a practice example before saying you're done. This will help you catch silly mistakes.

If you've reached your time limit without finishing the problem, read the answer. Understand and learn from the answer. For popular questions, there are also YouTube tutorials explaining how to approach the question.

If you notice yourself avoiding certain types of questions that probably means you're not comfortable with them. This is a sign to spend extra time on these concepts. Confront your fear of these problems. You might not be confident in your abilities to solve certain types of questions right now, but you can improve your skills.

From what we've noticed there are patterns that can be found in the solutions of certain types of coding questions. Once you've done a few, it can get easier to solve them. Finally, remember that consistency and practice are key. Don't beat yourself up for not knowing how to solve a question. It's normal to not know how to solve questions and you will absolutely improve with practice.



# INTERVIEW PREP

## Mock Interviews (CRITICAL!!)

Solving a problem alone on your computer is NOTHING like walking a stranger through your thought process as you solve a question in an interview. Communicating your thought process aloud is HARD. The first time you try a mock interview, you may be like a deer in the headlights. It's totally normal. It took us many mock interviews to become more comfortable explaining how we approach coding problems.

Gather your friends and practice solving questions together on a whiteboard. This experience will be invaluable once you're in a real interview. It even helps to practice in front of someone with no computer science knowledge. You can also use a website called Pramp that pairs you with a stranger to do online mock interviews. We personally don't recommend this site explicitly for the coding questions, but it's a good alternative to get comfortable and practice talking through your thought process to see where you're at. To be honest, sometimes this better mimics the environment of an actual technical online interview, since your interviewer will also be a stranger. The IDE setup is similar as well.

### \*Secret tip\*

Whoever is interviewing you most likely knows the answer to the question. If they start to look confused about what you're doing, this might be a sign you're not on the right track. Stop and talk to them. The best interviews are sometimes more like conversations where you work together to solve a problem.

## Tip on confidence

Believing in your ability to solve interview questions can be a challenge. Try to build confidence and warm up your brain with some easier questions before jumping into the tricky questions.

## Familiarize yourself with your resume

Be prepared to talk about anything on your resume. If you've listed a skill or experience on your resume, it's fair game for the interviewer to ask you about it. Being able to talk about your past technical experiences at a very technical level is important as that is what commonly makes up the "talking" portion of a SWE interview (ie. when you aren't coding). If you've written that Malboge is your favourite programming language, be prepared to crank out some awesome code in it. That being said, your resume is also your showcase. Don't be too humble, but make sure to refresh your mind with what's on it before your interview.

## Take care of yourself

Preparing for interviews can be an intense process especially for anyone juggling school, jobs, clubs, and personal life. Remember you still need to eat and sleep. Your brain will not function without food or on 2.5 hours of sleep. Also, try and not fall totally behind in school (easier said than done). You don't want to fail your midterms.

Finally, remember to take breaks. Whether it's sweating out all your stress at the gym or having a dance party in your room alone, don't forget to make breaks a part of your interview prep plan.

# THE ACTUAL INTERVIEW

## **Day of the interview**

You're ready and it's time to hype yourself up. This is not the moment to be cramming Leetcode hard problems.

On the day of an interview, you can spend about 30 minutes refreshing your mind on basic concepts or looking over a couple practice problems that you've already solved to calm your nerves. You could also go on a walk or run to get your blood flowing.

We would recommend preparing some questions for your interviewer. Remember they're interviewing you, but you can also interview them. They are employed at the company so they can provide insight on what it's like to work there. "What do you find most challenging about your work?" is one of our go-to questions. Having questions prepared shows that you're engaged and interested in the job/company.

## **The Actual Interview**

It's time to let your SOFT and TECHNICAL skills shine. Our biggest advice: talk, talk, talk. If your interviewer doesn't know what you're thinking, they cannot help you. By sharing your thought process and continually talking, they'll be able to better understand you and guide you back on track. It also helps to be positive and friendly. Remember they're partially evaluating you as their potential co-worker.

Additional things to keep in mind:

### **What do you do if you've already seen the interview question?**

Tell your interviewer. This helps foster trust. It can be really obvious when someone already knows the answer to a question and isn't taking the time to work towards the solution.

### **What do you do if you're being asked about a concept you've never heard about?**

If you're completely unfamiliar with a certain topic, it's ok! Tell your interviewer. Your skills cannot be evaluated on something you truly don't know. You might solve it together and be tested on your ability to learn instead.

Be prepared for the unexpected. Sometimes things just don't go the way you would like. Try your best, that's all you can do.

Please remember you do not need to answer every question perfectly to get an offer. Interviewers should be looking at how you approach problems, your communication skills, your technical knowledge, and not just your ability to think of the correct solution.

# AFTER THE INTERVIEW

## After the Interview

Follow up with your recruiter and give them a quick summary of how the interview went (what you and your interviewer talked about, how you felt about it, etc.). Be positive and polite. If something unexpected happened in your interview, this is a good opportunity to let your recruiter know. We would also recommend asking when you can expect to hear back with the next steps. \*Warning\* many companies have multiple rounds of evaluation in their hiring process.

Now give yourself a big pat on the back. Interviewing can be really hard, mentally draining, and a lengthy ordeal. Another tip throughout the interview process is to surround yourself with people also going through the SWE interview process. This way you can commiserate and prep together. BUT now it's time to treat yourself. Jam out to some tunes, indulge in all the sweet treats (cogro cake is always a solid option)- whatever makes you happy. And then, get a well deserved night's sleep without nightmares of coding!