Pathrise
PRESS

HOW TO LAND A

Software Engineering Job.

Intro

What is this guide?

This guide is a free resource to help students learn how to best approach their job search. We share many of the best practices we've learned and developed from working with hundreds of students that we've helped land their dream jobs.

Contents

What is Pathrise?

Pathrise invests in university students or young professionals by providing personalized advice and training on how to get the best possible internship or job. The program is completely free upfront and is entirely about optimizing your job search. This involves services like resume review, prospecting, referrals, interview preparation, and negotiation advice. We track every data point so we can hold ourselves accountable to actually produce significant and measurable value for our students.

Part 1: Find Opportunities

How to find opportunities

If you have a 5% chance of getting an offer from any opportunity. That might seem like a rough proposition, but mathematically that means that 50 such opportunities will give you a 90%+ chance of getting an offer. Part of your success will be dependant on the number of opportunities you have available, and if you don't get enough, it will be difficult to succeed.

Referrals

Referred candidates are nearly 15x more likely to get a position and nearly 80% of recruiters noted referrals as their best way of sourcing hires.

Tactics for finding people to refer you to opportunities include

- Ask fellow students from the student organizations you are in if they can submit a referral for you to the companies that they worked at last summer. Even if there isn't a formal referral system, you can have your friends send an email to their recruiter from last summer introducing you, and that will usually work.
- Parse your LinkedIn network and even consider connecting with employees at the company you're targeting, even if you didn't know them beforehand. If you connect and ask politely, many designers in the industry are looking to give back, willing to answer your questions, and would consider you for a referral even if you reach out cold.

- Utilize an alumni database provided by career services to find alumni at the companies you are interested in. Alumni are generally great warm leads and very open to helping students from their alma mater.

It's definitely possible that you won't be able to get a referral everywhere, and so there will have to be some positions where you apply on your own and use other tactics.

Career fairs

Career fairs can be a really great opportunity to network and connect with employers at your university, but don't just stand in random lines.

Instead, do some research before and review the list of employers that will be attending.

Ideally you have prepared a "why you are a good fit" for the high priority companies which you add to the end of your elevator pitch.

Go to the career fair prepared:

- Wear clothes that follow the career center dress code (If there is no dress code, wear clothes that make you feel confident).
- Come with multiple copies of your resume (40 is a good number).
- If you want to share an app or website you worked on, have it ready on your phone or iPad

Have your elevator pitch ready:

We talk about how to construct an elevator pitch later in this guide in the interviewing section. You usually want to do an abbreviated version of your go-to interview elevator pitch for career fairs (so 30 seconds to 1 minute rather than 1 to 2 minutes).

Go to the career fair and try to make sure you talk to as many companies as possible, and follow up over email afterwards as quickly as possible. Ask for the email of the person you chatted with and follow up - this is a very effective way of getting into an interview process.

Online

The best places to find jobs are your ABC's. These are job **A**ggregators, job **B**oards, and job **C**ollections.

Aggregators are usually a combination of a job board and an automatic scraper that collects jobs from elsewhere. They are the most convenient places to look for jobs: you can expect to find 5-10 viable opportunities in just a few minutes.

Boards are places where companies pay or sign up to list specific jobs. They are great places to look for positions that you might not find elsewhere, but you'll find jobs at a slower rate than when using aggregators.

Collections are extremely effective tools that essentially are manually curated lists of available jobs. Though sometimes they aren't always properly updated, for the most part they are a great way to "browse" through hundreds of individual company career pages at once.

Here are a few of the best examples of each of the categories:

https://www.google.com/ | Aggregator
Surprisingly good search tool, try to limit to only more recent jobs like posted in the past week

https://www.glassdoor.com/index.htm | Board
Also try to look under more unconventional locations if you can, different listings than Indeed

https://www.linkedin.com/jobs/ | Board

There are plenty of good jobs listed here

https://www.wayup.com/s/ | Board

University-specific jobs so that's great, but use the search function first

https://www.joinhandshake.com/ | Board

University-specific jobs and exclusive to each university.

http://www.intern.supply/ | Collection

For interns only but really high quality. Isn't always the most up-to-date

https://github.com/j-delaney/easy-application | Collection

Mostly only full-time positions, but pretty comprehensive list

Here are some other sample sites that are more for specific purposes or more general, but worth mentioning:
https://www.indeed.com/ | Aggregator, https://www.simplyhired.com/ | Aggregator, https://www.ziprecruiter.com/ | Aggregator, https://news.ycombinator.com/jobs | Board, https://angel.co/jobs | Board, http://jobs.gamasutra.com/ | Board, https://www.monster.com/ | Aggregator, https://jobs.github.com | Board, https://stackoverflow.com/jobs | Board, https://www.producthunt.com/jobs | Board, https://whoishiring.io | Aggregator

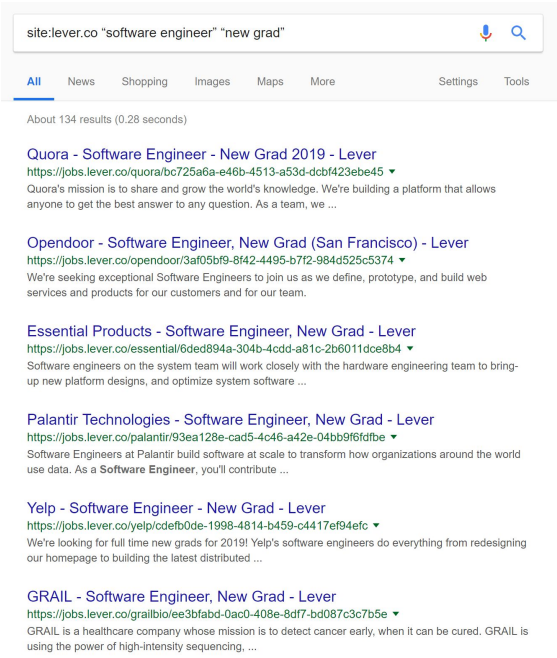
Ways to look

Another tool that is not used often enough is the X-ray search. An X-ray search is something that you can type into a Google search bar to find new opportunities. This way you don't have to

depend on your ABCs in order to find positions for yourself. Try typing in the following search string into Google:

site:lever.co “software engineer” “new grad”

You’ve basically created your own personal list of jobs. It should look something like below:



You can switch out terms as you see fit. The general anatomy of an X-ray search is this:

site:[insert the site you want to search] [keywords you want to search]

So other example x-ray searches you might want to try are:
site:lever.co “associate software engineer”
site:workable.com (software engineer AND “san francisco”)
site:greenhouse.io (“software engineering internship” OR “software engineering intern”)

These searches are very customizable and are a good way to create your own job aggregator or search engine and find positions that are more hidden away from other candidates.

Part 2: Optimize Your Application

Resume

How do recruiters read resumes?

Recruiters will spend no longer than 30 seconds on a single resume, and will routinely spend even less than 10 seconds reviewing most of them. This means your resume has to be easily skimmable and impressive at first glance, and this is a guide to help you optimize for those qualities.

General qualities of a good resume

Here are some of the general qualities of a good resume. Most of them are pretty self-explanatory, so we'll just leave them here with a short descriptor:

- **Pick a narrative** - Will help you decide what to keep.
- **Use keywords** - Use them in context if possible
- **Make every word count** - Don't waste space on unhelpful words; avoid passive voice
- **Lose the objective** - Oftentimes this doesn't add much information to a resume and just wastes space
- **Put the most impressive things first** - Maintain reverse chronological order for the most part
- **No typos** - Correctness is important for professionalism

- **Showcase projects** - Pick 2-4 only, 5 is pushing it but possible. Side projects outside of class are especially interesting.
- **Add contact information** - Name, number, email, website (optional).
- **Avoid too many colors and shades** - Most of the time, just do black with different font weights. If you are using color, 1 is more than enough.
- **Make sure the text is selectable** - Applicant tracking systems (ATS) won't be able to process it otherwise.
- **Keep your margins clean -** Generally 0.5" - 0.7"
- **Be smart with your GPA -** GPA under 3.5 is doubtful to list, definitely don't list a GPA under 3.0.
- **Spacing & distinction -** Make sure your section headers, item titles, and subtitles are visually distinct using different font styles. Also, keep spacing as consistent as possible.

The 2 biggest mistakes on most resumes

Almost every resume we review makes the same two mistakes or has the same two points where they need to be optimized. We go into detail on what these mistakes are and how to fix them below.

Mistake 1 - Grunt vs. Impact

Grunt is our internal reference for resume points that are focused only on what you worked on and what you were assigned to do. You're essentially describing the grunt work that you did, and it's usually never a good description of how you spent your time in any past experience or project. Grunt statements usually look something like 'Developed X for Y' or 'Worked on X using Y'. Grunt statements are sometimes a necessary evil, but for the most part should be avoided. Instead…

Impact statements are statements that focus on what you have accomplished and what your results were. They normally follow a structure that's more like 'Accomplished X by implementing Y which led to Z' or 'Developed X to accomplish Y and Z happened'. They might sometimes be a little bit longer, but being able to show some reading your resume why what you did actually mattered is definitely worth it.

Mistake 2 - Lack of Quantification

Students often struggle to find numbers because they feel like their projects need to be launched and successful to find numbers, but this couldn't be further from the truth. There are so many statistics that you might be able to list about your experience and projects and how they went. It's just a matter of knowing where to look.

In order to find numbers where there may seem to be none, you should ask yourself questions like:

What was the scale of this problem?

- How large was my dataset or many rows of data did I analyze?
- How many devices did I serve?
- How many scenarios/permutations/tests did I consider/handle?
- How many different methodologies did I implement?
- How much more time did I invest than is normal?
- How many people did I manage or how many teams did I act as a liaison for?

What did I achieve as a result?

- How many users did I launch to or will I launch to?
- How many users/groups used it?

- How much money did I produce in value?
- How many hours did I save the company?
- What percentage did I improve our old process?
- What percentage of our old process did I replace?

Generally, you will find that coming up with these questions is more of a matter of understanding the general gist of where to look rather than some magic sauce. Hopefully the above examples give you a better idea of how to get started looking to quantify your resume points.

Below, we've included some examples of mistakes of both grunt instead of impact and lack of quantification, and how students applied the concepts above to fix them:

Note: The following examples have been slightly edited to obfuscate identifying information

Before	**After**
Created a larger social media presence by blogging on the company website and writing ads that promoted housing options.	Spearheaded social media and advertising efforts, creating leads that contributed to selling 30+ properties worth $20M+ in total.
Generated KPI's (number of blockers, average resolution time…) on the integrated data through a Java web application using Spring/Hibernate.	Developed a Java web-application using Spring/Hibernate that reduced JIRA task resolution time by over 25%, meaning that overall company efficiency increased by 25%.
Originated concept for social network app for campus organizations.	Built a social media platform for campus organizations, successfully launching to ~15 clubs and aiming to have 200+ clubs by Spring 2018.
Aided undergraduate students with understanding of code, homework, project, and topics (including data structures, algorithms, and Java Swing).	Taught personal office hours for 200+ hours, going above and beyond by investing 3x the average expected teaching time expected of a TA.

Reverse recruiting

One of the best ways to add on to your application and make it more effective is to send an email along with it. This will get recruiters to look at your resume a second time and increase your response rate significantly. This is what we like to call "reverse recruiting" because instead of waiting for the recruiter to reach out to you, you reach out to them first. According to our data, sending an email along with your applications will increase the likelihood of a response by an average of 3x, which is a huge benefit, especially since sending an email can take less than 5 minutes per application.

For every application that you do, search for someone on the marketing or recruiting team and send them an email. There are many ways that you can find someone's contact information. On one hand, you can see if their contact information is available via an email-finding tool like the ones below:

https://connect.clearbit.com/

https://www.circleback.com/contactcloud/

Alternatively, you can find somebody on LinkedIn and try to guess their emails. The most common patterns are:

first@company

firstlast@company

flast@company

first.last@company

f.last@company

first.l@company

After guessing and before sending, you can use verifiers or automatic email guessers like the following sites to verify if your email is correct or not, so that it won't bounce when you send it:

Hunter (https://hunter.io/email-verifier)

LeadFinder (https://www.leadfinder.pro/)

Email Checker (https://email-checker.net/)

If the company is small enough (under 100 people), you can even reach out to the CEO or CMO, and sometimes they'll reply, and maybe even interview you themselves. That's the sort of attention that you can receive if you ask for it in the right way.

But what is the right way of asking for it? A good cold email has the following 5 features:

- **Concise** - Short and to the point.
- **Compelling** - Mentions something about you and your interest in the company that is convincing enough for the person reading it to respond. Does it excite them?
- **Personalized** - Doesn't seem overly templated and seems like you wrote something unique for the particular company.
- **Friendly** - Comes across as friendly and casual. Usually you can be more relaxed in the technology industry than in some other fields like finance.
- **Correct** - Avoid typos, misspellings, and grammatical errors.

Below is an example of a high quality outreach email that illustrates all of the above principles from X who is passionate about joining Discord as a growth marketing associate.

Hi Gaby,

I'm X, a senior at San Jose State University majoring in Computer Science and Business graduating at the end of 2019, and I'd love to learn about how I could contribute at Discord.

I've been using Discord for more than 3-4 years; from day one I was advocating for all my friends to quit Skype. Surprise! It happened. Seeing how much Discord has improved and how many amazing features

have been added over the years has made me super proud. Since then I've always been interested in designing awesome things at Discord myself.

I know you're super busy, but it would be awesome to be able to hop on a call with you to chat about what software engineering roles you and the Discord team would be potentially hiring for in the near future. Would you be free for a 15-minute call on Monday either at 1:00 pm or 3:00 pm (PST)?

Just in case, I've attached my resume to the email for reference and my resume can be found here (link). I appreciate your time and consideration!

Regards,

There are many ways that you can adapt the cold emailing strategy above to do even more interesting tactics.

For example, if you want to switch up the platform, you can use the following features above to write a strong Twitter message to a software engineer or engineering manager you admire from a target company and see if they'll be interested in you as a candidate or for an initial chat.

If you want to switch up the purpose, you can use the following tactics above to email someone before you apply to a position, so that you can set up a conversation with them to learn more from their career and form a mentorship relationship. Afterwards you can follow up politely asking them for a referral if they're comfortable with it and get your foot in the door at a company that way.

With creativity and the building blocks above, the variety of tactics that you can apply with reverse recruiting are endless!

Part 3: Nail Your Interviews

Interviewers are going to ask you questions about fundamentals like algorithms and data structures. They want to test the tools in your toolbox. With stronger basic skills, they'll know that you will be prepared to learn any new framework or technology.

Along these lines, there are a few keys to understanding how you want to prepare for technical interviews:

- The first thing to do is to master a single language. Most top companies want to know if you're good at coding but don't care as much if it's in one language or another. While you don't want to choose a language that makes your life harder, like C, and generally want to pick a language with some functions pre-built for convenience like Python and Java, at the end of the day, it's up to you.

- Next, you want to make sure you are good at debugging. You will need to do it plenty of times throughout the interview.

- Finally, you need to get comfortable with thinking out loud and vocalizing your thoughts. It makes a huge difference if your interviewer can actually follow along. An interview without a solution, but with thinking out loud, is much better than an interview with a solution in complete silence.

Remember that almost all of your questions will require understanding of one of the following:

- Hash tables
- Linked lists
- Breadth-first search
- Depth-first search
- Quicksort
- Merge sort
- Binary search
- 2D arrays

- Dynamic arrays
- Binary search trees
- Dynamic programming
- Big-O analysis

So study them hard and commit each of these fundamentals to memory.

How to be most effective when prepping for technical interviews alone

The goal is to simulate the environment as if it was a real tech interview. We recommend the following guidelines to make the best use of your time while prepping for technical interviews.

Use pencil/pen and paper (landscape orientation)

This simulates the whiteboard environment the best. If you happen to have a whiteboard, use that instead! If you are rusty on syntax for your chosen language or are using some new semantics that you roughly recall but don't know the exact details, we recommend the following:

- While actually coding the question, just attempt to write the best pseudocode possible for the portions of code in which you forgot the exact semantics. For example, if I forgot how to initialize an integer array in Java with values I would just write:

```
int pseudoCodeJavaArray = [1,2,3,4,5]
```

- Once you're ready to execute the code in a programming environment, it's fine to look up the right semantics so your code can compile/run. We recommend you have another document where you record all the semantics you forgot and what's the best/correct way to write them. So continuing with the example above you would note down:

```
# Initialize an int array
int[] myIntArray = {1,2,3};
```

Stay completely focused for 90 minutes

Block of 90 minute times on your calendar where you do nothing else but attempt the question (~20-45 minutes) and then review the answer afterward (~30-45 minutes). Remember, we're trying to simulate the interview environment/pressure as much as possible. You wouldn't check out your Facebook timeline if you were having a brain fart in the middle of a real interview, right?

Consider this a time limit for completing the question as well and even try to finish most questions within 45 minutes at most. This will help you simulate the actual speed required in a real technical interview. You can use the latter 45 minutes for review like mentioned above.

Speak out loud as if you were communicating to an actual interviewer

Speak out loud for the parts that you would be speaking in a real interview. Some guidelines are:

Problem solving stage

- Talk through the solutions you are thinking. Discuss tradeoffs between solutions if you have many approaches.
- Always state the runtime and space complexity of each solution you come up with.
- If you are struggling to come up with a more optimal solution, talk out loud about your thought process.

Coding stage

- It is not necessary to talk about every line of code you are writing. Interviewers know it is difficult to talk and code. A good balance is to let the interviewer know when you are doing for a logical chunk of the work. For example: "I'm going to create two loops that iterate over the array and sum the unique pairs." However, do whatever you feel most comfortable with here.
- It is also totally fine not to talk at all as long as you are making progress on the code.
- If you are struggling for more than 2 minutes to come up with the code to solve particular state you're trying to make, talk it through out loud. A common question interviewers will ask is:
    - "What are you struggling with?"
    - "Can you tell me what you are thinking about?"

Verification stage

- Tell the interviewer about which edge cases you are going to test.

- When you catch a bug, explain to the interviewer what the bug is and how you are going to fix it/what you fixed.

You can even get novel with this! One candidate told us he talks to a basketball and uses that as a prop for the interviewer.

When do I start coding?

- You should spend at most 10-15 minutes problem-solving. Then, with the most optimal solution you have thought of, attempt to code. If you have two solutions that are almost equally optimal, code the solution you feel more uncomfortable with.
- By all means, if you already are confident you know how to code the solution (i.e. the naive solution), there's no need to waste time practicing what you already know. Spend more time problem-solving!

What happens after I'm done coding?

- Once you come up with a solution that you mentally verified is correct, don't just look up the answer!
- Type the code into an environment where you can compile/run it. Make sure it works in all the reasonable edge cases.
- Please read the "Don't Get Print Statement Happy" point in the coderpad.io section
- When you catch a bug, explain to the interviewer what the bug is and how you going to fix/what you fixed.

We have also compiled 93 software engineering technical interview questions that you can use to prepare.

Behavioral interviews

The first interviews in many processes will often be behavioral and we'll detail how to nail those below.

Answer structures and guidelines

When in a high stress situation it can be difficult to effectively organize your thoughts. That is why having a couple of answer frameworks memorized and ready to go is critically important. There are dozens of these frameworks out there for behavioral questions, but we have found that the following have been the most useful especially in marketing interviews.

Elevator pitch

The elevator pitch is a classic pitch format named after a hypothetical situation where you find yourself riding the elevator with the CEO or an investor and need to quickly pitch them an idea. Your personal elevator pitch should sell the most attractive parts of you as an employee. This pitch should be ready to go at a moments notice and briefly summarize the most impressive points about your education, experience, projects, and end with a summary of yourself as a candidate as well as a preview of why you want to work at company X. The structure looks something like:

Education: Introduce yourself, your major, and your class or year of graduation, which is really important for the recruiter to understand so they know what type of position you are looking for.

Experience: Talk about the past work that you've done in previous internships or even student organizations and activities. Show that you are not just somebody who learns in the classroom and doesn't do anything with it in outside activities.

Projects (optional): If you would like or if you don't have much experience, supplement your elevator pitch be mentioning 1 or 2 of your most interesting projects.

Conclusion: Make sure to end strong. Don't just trail off when describing yourself. End with a brief reason why you are interested in the company you are pitching or end with a summary of your strongest skills and high level background.

Generally, you will want elevator pitches to take 1-2 minutes in an interview and 30 seconds to 1 minute in a career fair.

STAR method

The STAR method is a pretty common way of answering general interview questions. You don't necessarily have to follow it exactly, but it will help you give easily understandable and compelling answers:

Situation: Describe the premise of the situation or problem. What needed to be accomplished and more importantly, why was it important?

Target/Task: What goals were you working toward or what is the definition of your project?

Action: Describe the body of your work and the actions that you took to solve the problem or achieve the goals at hand.

Result: What was the impact that you made and how did the team or users react? Why was this experience either a great learning experience or a resounding success?

Generally, you will want your answers to be around 2-3 minutes, but this varies greatly, and it is better to give a full 5 minute answer as long as it is focused rather than an incomplete answer.

Keep your answers succinct

You want to keep your answers focused on the specific story or topic at hand. If you have an interesting tangent, you can indicate that by saying something like "I'd be happy to add more detail to this later" and continue on with your main answer for now.

Be specific

Describe concrete steps that you took to solve problems and what tools you utilized. Talk about the tools you used, the types of questions that you asked, or the kind of solution that worked.

Mix "I" and "we"

When it comes to taking credit, if you were on a team, use "I" to denote what your responsibilities were and be very clear and then use "we" to give your team credit for overall accomplishments. Alongside that, mention how grateful you are or how much you learned, to avoid sounding too much like you're bragging.

Avoid negative language

This is an opportunity for you to talk about the great work you've done. Even if the project wasn't launched or it didn't have the desired result, you don't need to use negative language in your responses. Avoid overly pessimistic words like fail and disappoint, critical words like stupid or idiot, and self-degrading words like lazy or clueless.

Before the interview

Research the company. 15 minutes of research can have a serious impact on your ability to relate yourself and your experience to the company. The “About” and “Product(s)” pages will often have all the information you need. Look for the company’s mission statement, their history, and read the descriptions about their core products. If you’re looking to go above and beyond, you can also dig into their blog and get some good bits of information from there as well.

In these interviews you will be evaluated based on how well you fit the cultural values. These can typically be found on a company’s “About” page on their website. Shaping your answers to fit their values is critical. Also of incredible importance is effectively communicating why you are interested in the mission of the company and you certainly can’t do that if you don’t know what their mission is. You can typically find information about the company mission on their “About” page as well.

The end of the interview

Ask thoughtful questions to show how interested you are in the company and that you’ve actively been thinking about the position, their products, their mission, etc. We’ve come up with the best questions for software engineers to ask in their interviews to ensure they show interest.

1. What are your current challenges on the team?
2. How can I, in the role that I’m interviewing for, best contribute to solving these challenges?
3. What opportunities will I have to learn new languages or technologies, related to the work that you are doing here?
4. Is the engineering department more team-based or autonomous?
5. How much guidance can I expect from my supervisor and how much ownership can I expect for my projects?

6. What are your favorite aspects of working at [company]?
7. What is the most difficult part of your day at [company]?
8. How can my work directly affect the company's mission of [company mission]?
9. I read that [recent update about company]. How does that affect [the mission, the work of the engineering team, etc]?
10. If I were to start on the engineering team tomorrow, what would my first task be?

After the interview

Be sure to follow up with your interviewer with kind, grateful message, however don't overdo it here and risk coming off as not genuine. This follow up is also a chance for you to correct any mistakes that you may have made in your interview. Mentioning how you continued to think about questions or problems that they asked and suggesting corrections to your mistakes can be the difference between a "yes" and "no".

Conclusion

These are just the basics

There's so much more to talk about when it comes to best preparing for your job search. We've covered some of the basics to get you started above but to give you an idea of what else there is that the Pathrise program covers, including but not limited to:

- Portfolio websites
- Project development
- Career path case studies
- Advanced resume optimization
- Informational interviews

- Cold email retroactive templating
- Boolean searching on LinkedIn
- Preparing for high frequency behavioral interview questions
- Building a behavioral matrix
- Advanced technical interview questions
- Company specific interviewing frameworks
- Mapping out interview processes
- Understanding level/geo compensation codes
- Negotiation frameworks
- Modeling job search pipeline probabilities

The job search is already an incredibly stressful process and our intention with this guide is certainly not to overwhelm you. Our goal here is to help you understand that the job search is something that should be approached systematically and optimized. The more effort and thought you put into it, the more you get out of it. So don't just apply to a few places and hope to hear back. Take your destiny into your own hands and start being proactive with some of the tactics that we've shared with you above.

Good luck and best wishes from Pathrise!