

Resume Workshop

@TechLead (Ex-Google Tech Lead)

I've reviewed hundreds of engineering resumes. Here's a recap on some of our best tips below.

Resume Templates

Check out my sample resume to get started.

- [TechLead resume](#)

The 10-second rule

Take 10 seconds to scan your resume from the beginning, reading through it quickly (as if you had a dozen other resumes to get through). This may involve reading only the first half-page, scanning the first 5 words of each sentence/bullet point.

What do you see? If it's all over the place with fluff words, weak verbs, and shows weakness, chances are your resume will get thrown out because 10 seconds is probably about as much time as a recruiter/hiring manager is going to give you.

Here's what you need to ensure:

- The first words at the top of your resume should be very *software-developer* oriented, around *developing, implementing, creating, and building*:
 - **"Developed** a REST API in X using Python and Java..."
 - **"Built** a full-stack web app and launched to 1000 users..."
 - **"Taught myself** GoLang and **implemented** a framework for a webserver..."That's starting to sound like a candidate we might be interested in.
- The resume should flow in sentence structure, explaining one after another in active verbs what you have personally done. There's a rhythm here: *"Built X, Created Y, Developed Z, Implemented A, Built ..."* If you are breaking up the resume with statements rather than actions like *"GeoPickV3: A project to terraform the geoclusters,"* that halts the **readability** of the resume with fluff words that explain absolutely nothing, and switching the subject between you, your project, your team, and your company causes mental breaks. The subject should be *you*.
- It should be clear that you are a strong developer for the targeted role within the first few moments of reading your resume. If it is not strong, you need to rephrase your resume.

Pitfalls:

- Don't use non-engineering words like "Documented," "Supported," "Troubleshooted," "Automated unit-testing," etc., You are not applying to be tech-support here or a documentation writer, so there's no need to highlight those points. Just like how you wouldn't highlight that you can cook, even though the company may need a few good cooks – that's not what you're expecting to do here.
- At the top, don't list low-priority information like hobbies, extracurriculars, or irrelevant skills like Photoshop, Premiere, Excel. Move these towards the bottom of the resume, or just leave out skills that may dilute your resume.
- Don't spend too much time explaining what the company does or what the team does. Explain what you personally did.

Tell a Story

Descriptions should tell a **story**.

It is absolutely useless to list a bunch of technical jargon that no one knows. Imagine reading a laundry list of:

- Implemented X (version 238200)
- Implemented Y in SprintSpark 3
- Implemented Z using GeoLoaderXX1

It just doesn't matter what someone implemented if there is no story behind it.

Even worse, many people don't explain the languages used, and these technologies are often all proprietary so there is no way to reuse the skillset.


Saying that you "*Implemented Carraform v2.0 and launched to Prexo SpringSuite*" is like saying nothing, and yet most resumes are absolutely littered with phrases like this.

I would recommend you explain your descriptions in a format of
[ACCOMPLISHMENT, IMPACT, CHALLENGE]

This 3 components tell a complete story. For instance, you might say "*Developed a server-side layout engine in iOS by teaching myself GoLang to automate 1000+ antiquated manual layouts. Working with a coworker, this began as an ambiguous project that I drove to completion and launched company-wide.*" This is far more impactful and relatable, since it explains what you did, why it was hard, and what the validation of that work was.

Less is more.

Focus on a few very strong points and develop those, instead of a long list of bullet point features, none of which seem very interesting for anyone to care about. If you have a big



accomplishment, spend more time on your resume to talk about that. The fact is that any bullet point could have an amazing story behind it, but you need to reveal that story.

I once saw a resume with a "Co-founder" role and yet there was a single 5-word bullet point. *"Co-founded a company and launched X."* That looks... disingenuous. Was this real? What were the challenges and impact? Is this worth talking about? Obviously not if the candidate didn't have much to say about it.

It's about you. Not about your accomplishment

What many candidates fail to realize is that I want to know what **you**, as an individual, did.
Not what your *company* did.
Not what your *team* did.
Not what your *project* did.
*What **you** did.*

Because the fact is, the only thing you're able to bring to my team is **you**.

So I want to know about your personal contributions to a project. That's it -- what type of character are you, what skills do you have, how did you apply those skills, what motivates you. It doesn't matter if your company is *"the worldclass leader in B2B enterprise solutions with 50,000 clients,"* if the only task you did was write tooling documentation.

So, don't worry so much about listing every little accomplishment like how you *"Implemented Terraform in CarbOS using Carabide."* Nothing in that sentence is usable or translatable. But if you were to say *"Taught myself Carabide in 2 months and collaborated with 3 teams to lead development of Terraform, launching with CarbOS..."* That tells me a lot more about your ability to learn, your collaboration skills, and your passion for creating.


Show. Don't tell.

Every bad resume seems to have a portion where the candidate will say:

"I am hardworking, enthusiastic, and passionate. I am a great communicator, have excellent teamwork skills, and am perfect!"

You're telling me, but you're not showing me. What reason do I have to believe you?

If you're such a great communicator, you should have the stories to back that up -- instead of telling me that you can communicate, show me that through a story about how you *"Mentored your team through weekly presentations, leading to the adoption of the ViewModel pattern in the codebase."*



If you're passionate, tell me about a pet project you started.

If you're hardworking, tell me about how you taught yourself a new language and shipped it under a tight deadline.

The resume is where these stories paint a picture of you.

It is your personal marketing brochure, not a fact-sheet.

What in here isn't BS?

A trained resume reviewer will be scanning for fake, BS stories. For a resume to stand out, it needs to contain **validation**. You validate your skills by explaining the results – the impacts of your work. Some tips here:

- For each story, list the impact. Instead of saying "*Implemented X*," say "*Implemented X, and was featured in the TechCrunch.*" Or "... *this work was adopted to the wider team.*" Or "... *saving \$XX thousands of dollars,*" or "... *and sped up developer velocity.*"
- I've seen resumes with 10 pet projects, and yet not a single link to the project website or github or anything. Where's the evidence of any of this? Were these worthless projects that were trashed?
- Put in concrete technical details where they count to substantiate your claims. If you have a one-liner "*Developed the iPhone,*" that's just not believable. But if you can expand on that with all the relevant technologies & challenges, that makes your claim believable. Grounding the claim in reality makes it impressive.

People won't click on your links.

A lot of candidates put links in their resume, which is great. But just know that 90% of people will not be going through the hassle of opening each and every link.

I've seen a number of resumes where people say "*Built Project Maven (github.com/myproject).*" The description here isn't sufficient and it needs to stand on its own - you can't assume anyone will go to these links.

- If you have links, it may help to have a single personal page (*yourname.com*), rather than a dozen links to your github, linkedin, facebook, twitter, personal page, project page1, project page2, etc., Who's going to open all of that?
- Make the links easy to type, using a shortener or custom domain.

- If someone takes the time to open any of these, make sure the projects look good. If I waste 5 minutes of my time to check out your personal GitHub, and it's just scrap waste projects... guess where your resume's going too?

Target your resume to the role.

A lot of candidates get confused about "facts" vs "story". *Your resume is not a fact sheet.* It is a targeted story, subjectively told in an effort to persuade a biased decision.

If you worked in a "Hardware Engineer" role and now you want to do Software Engineering... do you talk about your hardware work? If you can cook, do you talk about that? If you worked as an Uber driver, do you list that? The answer to all of these is "no."

Depending on the role at hand, you want to highlight just the relevant skills. Even if at your last job 10% of the work was Python, if you're applying for a Python role you want to be gushing about all the Python development that you did. *"Implemented X in Python. Used Python to do Y. Taught myself Python frameworks Django and Flask to do Z."*

If the role is for Machine Learning, focus in on those keywords.

If it's for a web frontend engineer, focus in on your web javascript skills.


If you aim for everything, you'll hit nothing.

You must code.

The hilarious thing is sometimes people apply for SWE roles and yet they don't seem to want to actually code. They'll go into the interviews and say:

- I don't code anymore. I'm more into strategy.
- I lead projects.
- I managed my team and coordinated them.
- I'm a co-founder and CEO.
- I'm an entrepreneur.
- I have leadership skills.
- I'm a great communicator and team player.

I once saw a resume where the top 3 experiences were "Co-Founder", "Co-Founder", "CTO." OK, maybe you should go apply to be a cofounder then? Because this role is for **coder**.



Too many people don't want to code, or think they're "above the code." Many resumes I've read seem reluctant to mention code or languages. People might say "*Created X, Y, and Z and shipped this project, leading and coordinating development on X.*" Well, where in this sentence did you actually do coding?

Let's put it this way: If you've applied for a SWE role, you're expected to code. Not to waltz around the office in a suit going to meetings with the VP of Sales.

So show the languages, frameworks, and technologies that you're using.

A lot of people will bucket all of that into the bottom of their resume under "*Skills: C, Python, Javascript*" and never mention any of these again in the rest of their resume. That is simply **not** very convincing evidence that you can actually code. It doesn't explain how well you can code.

Formatting

Let's talk about formatting. It matters.

- **Single page.** If you are using more than a single page, you are rambling. Trim it down because no one has time to read all of this stuff. Highlight just your best, most recent work like a brochure.
- **Single Column.** Double-column layouts and fancy fonts/graphics generally just distract and people can accidentally skip portions of your resume. It doesn't flow.
- **Fully spell the names of companies.** Which looks better? "**VMWare**" or "**VMWare, Inc.**"? The mental weight of the fully-spelled name makes it look more official and professional.
- **Fully spell the names of your degree & education.** Similarly, don't short-change your 4-year degree (if you have one) by saying "*Majored in cs at Ohio state.*" Make it look good. "*Bachelor's of Science in Computer Science. Ohio State, 2004. 3.80 GPA.*" That's better. It may even be appropriate here to list additional activities, honor societies, relevant coursework & projects to bulk up the education a little more so that it's not just a one-liner.
- **Grammar check.**
- **Spelling check.**
- **Capitalization check.**
- **No photos.** In the US, it is illegal to discriminate based on age, gender, race, or appearance. Don't put photos on your resume for that purpose.
- **Your name.** I've been toying with this idea recently, but it's something to think about. Let's pretend you have a name like "Xheycho Yining Sundarabai Xylophone." I'm sure it's a beautiful name in your language, but it doesn't look familiar for many people. I've seen some colleagues with unpronounceable names and they say "Just call me Bobby," or "Just call me Shay" and that instantly seems to make them more relatable. Think about it -- your name is your personal brand and is to be memorable. It paints a picture about

the person behind the resume. It may help to put something like: "Shay" (Xheycho Yining Sundarabai Xylophone). (Up to you though, this is just a theory).

- **Age.** Age discrimination is absolutely illegal. If age is a concern for you, consider removing your graduation dates. No one will ask you to list those, because it is illegal.

The objective.

Not everyone needs an objective.

But, if you have a broad background, an objective may help focus your resume and clarify your intentions -- especially if you're switching fields, people may get thrown off. An objective is a quick way to briefly explain what's going on here, instead of allowing a person to read the resume once, become confused at your history, then read it again, then think about the open roles, then try to figure out if there's any mistake here, etc.,

Oftentimes, I'll see a candidate's resume and it's all over the place. There's a little of everything. So I ask myself, *"First of all, what type of role is this person going for? iOS? Android? Backend? Frontend? Machine Learning? Graphics? Are they even interested in graphics?"*

This is one reason you need to focus your resume, as mentioned above. An objective is a tool for helping you do that if you feel your background is wandering all over the place.

Last Impressions count.

Every resume has a first impression.
And every resume leaves a last impression.
This is the last thing on there.
Make it look good and make it count.

I've seen some resumes that just end with something forgettable like:

Education

bs in biology at Canvers Institute, 2008

Does that look good to you? bleh.

I think even if you were to list a few fun facts about you as a person like a cool hobby, a cool project, or next to your education list some neat clubs/activities/coursework you did that can make your resume more memorable.

Make yourself seem like someone people want to bring in to personally meet.

Exercise 1

Create your resume

Tip: If you have a hard time with a template, checkout the Tech Interview Pro Facebook group for some samples.

List out your best accomplishments that are related to software engineering. Accomplishments of the following:

- Education/Degree and Relevant Course Content
- Programming languages you know
- Internships and Job Experiences
- Software Engineering Projects
- Competitions that are technical in nature

After you've created your resume. Send it to at least 5 friends or colleagues that have experience in the technical field. Let them critique it and modify your resume at their discretion.

Post your resume on Tech Interview Pro group to get further critique. Getting your resume right is extremely important, it's your most valuable asset for the first obstacle, which is getting an interview.

Exercise 2

Build a few web projects


If you feel like your resume is lacking experience (we will tell you), then you will have to build a few web projects. Start with a classic REST API + Web Front End project. With that fundamental understanding on how to build a web app, you will make your resume a lot more solid.

Here are some example tutorials online you can follow and build your own web app.

Example Tutorials

1- Build a RESTful API Using Node and Express

<https://scotch.io/tutorials/build-a-restful-api-using-node-and-express-4>



I highly recommend this tutorial. It will teach you how to build a backend that is compatible with most front end. The MEAN stack is extremely popular nowadays. However, this does not teach you how to build your front end. You can build any kind of front end that calls your API server.

2- REACT JS TUTORIAL by LearnCode.academy

<https://www.youtube.com/watch?v=MhkGQAoc7bc>

This series is extremely helpful to build your front end using react js.