Blogs

Writing your computer science resume

Read, research, and explore more to choose what's best for you! Let us know how we did @ 30dayscoding@gmail.com. Good luck, you'll do great!

Templates

- Career cup: What a GOOD Resume Should Look Like
- Overleaf: Jake's Resume Template (preference)
- Github: Deedy-Resume Template
- It's perfectly acceptable to have a simple resume. Nothing fancy is required - No images, colors, fancy fonts, headings, quotes, etc.

Action words

- Make sure every line starts with an **action** word:
    - Eg: Worked on, Created, Added, Implemented, Discovered, Generated, Collaborated, Researched, Managed, etc.
    - 240 Resume Words: Action Verbs
    - 185 Powerful Action Words

Work experience

- Ideally 3-4 experiences with 2-3 points each.
- Make sure the dates are accurate and consistent in style. Include the year and the month or season.
- Common headings: software engineer, web developer, mobile developer, intern etc.
- Add a line at the end for Tools: Angular, Flutter, Dart, Testing, etc.
- Don't add things like: using VSCode, Vim, etc. No one wants to know your code editor preferences.

- You CAN add experiences outside of coding/computer science if you learned something new and have space for it. It's better to show real experiences than nothing!
- Examples
    - Added *some* features using frontend technologies, which eventually went into production.
    - Worked on something which brought the revenue up by some %.
    - Implemented login functionality with authentication service using Firebase and Node JS.

Projects

- Ideally 3-4 projects with 1-2 points each.
- Use school projects if you don't have any personal projects (start working ☺).
- Mention the skills you learned or used. Eg: HTML, CSS, JS, Git.
- It's fine to add other projects as well - maybe you did marketing for someone and helped them increase their numbers!
- Examples
    - 30dayscoding: *Made a website for computer science students preparing for coding interviews which includes an all-in-one technical interview guide.*
    - *Created a landing page for the website to generate initial traffic before launch.*
    - *Worked as a freelancer to create 3 pages using something, something.*

Education

- A lot of people argue that education should come before work / projects. That's a myth.

- Add your degree, courses, clubs, activities, experiences, projects, etc. You can mention jobs you did at your university.
- Eg:
    - *B.S. in Computer Science, 2020*.
    - GPA: doesn't matter.

Skills

- Add programming languages, tech stacks you've used / are learning.
    - Languages: Python; Java; Dart; JavaScript; Typescript.
    - Technology: Android; Flutter; React Native.
- It's fine to *fake-it-till-you-make-it* when submitting your resume, but make sure you know it before your interviews.
- You can also add soft skills - they can help the resume parse through the ATS software. Eg: communication, leadership etc.

Summary

- Don't add a summary section unless you have space for it. Most resumes pass through an ATS software before being viewed by someone, so don't waste time on a summary.
- Generic things can be avoided most of the time. Everyone is a 'hardworking software engineer with excellent skills', don't add it here. Instead, add it in the skills section (if you have space).

Links

- Add links to your Github, linkedin, Angel list, Twitter profiles.

- A lot of times. managers and recruiters who look at these profiles will directly contact you, so it's *nice-to-have*.
- Github: Add school projects (with permission) if you don't have any personal ones, or just fork other repositories that you like. Keep exploring!

Harsh reality

- Only a limited number of people look at your resume!
  - Recruiters or managers screen it after you've cleared your first round of online coding interviews.
  - Even then, they will ask you to explain most things - 'what's your favorite project', or 'which tech stack are you most comfortable in' - they want **you** to explain it.
- Don't put in a lot of time.
  - It's fine to have a decent resume, but preparing for interviews or actually making projects is far more important.
  - Apply to a limited number of companies, prepare for those, and focus on clearing the interviews you get. Prepare well!
  - Getting a first round interview from a hundred companies <<<< Getting an offer from 1 company after applying to 10.
- Don't pay someone to do a resume review.
  - Ask your friends.
  - Ask someone in your university - consultants, advisors, professors, etc.
  - Lookout for free resources like this one.
- Your resume is only looked at for 5-6 seconds.

    - Make sure to keep that in mind when displaying your skills. Use actionable items like numbers, percentages, etc.

Don't follow this blindly

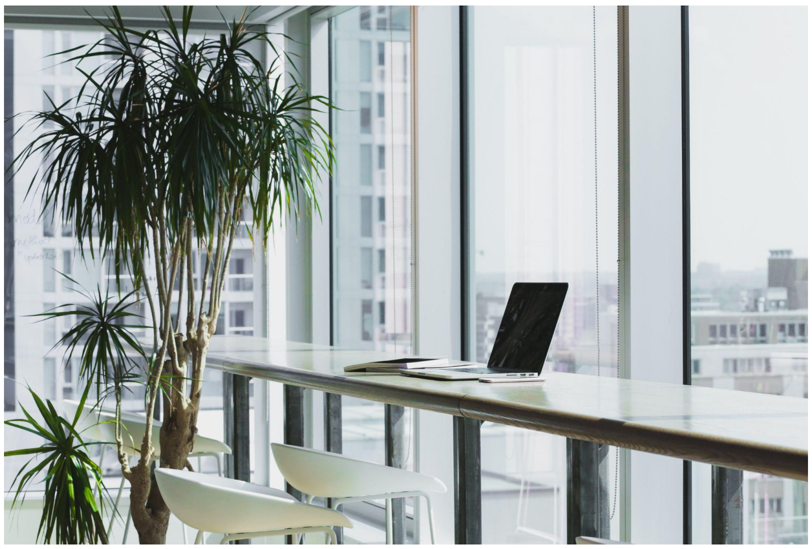
- This comes from our personal experiences, so don't take it too seriously, and feel free to make tweaks according to what you think is the best. Everything doesn’t work for everyone, so experiment along the way.

Good luck

- Your resume doesn't define you, you're worth much more than that. If you're looking for a job/internship - we hope you get it soon! Keep working hard. Checkout 30 days coding if you're preparing for coding interviews.

How to apply for Internships and Jobs

Read, research, and explore more to choose what’s best for you! Let us know how we did @ 30dayscoding@gmail.com. Good luck, you’ll do great!



Cold emails

- Cold email == emailing someone you don't know. It's the best proven way to get in front of someone's eyes directly and be instantly noticed.
- An example of this is sending emails to hiring managers or recruiters, telling them something about yourself and discussing a position/referral.
- **Selling point**
    - They get thousands of emails every day, so it's important to highlight what's unique about you! It doesn't have to be too fancy, but something that you feel strongly about.
    - Your unique 'selling point' can be anything - github profile, work experiences, projects, hackathons, projects, grades, certificates, courses, etc.
- Template
    - Here's a short blog on writing cold emails: Cold Emailing, the right way

Cold email Flow

- Find the relevant person on linkedin - recruiter, hiring manager
- Send them a message on LinkedIn
    - *Hey recruiter, I'm interested in the xyz position at your company! My past experience at abc would make me a strong candidate for this position, I've attached my resume and look forward to your response!*
    - More examples in the cold email blog here: Cold Emailing, the right way
- Find their email
    - Use LeadLeaper.com, ContactOut.com, PIPILEADS.com

- Ask your friends, teachers, or relatives if they know people at these companies and email their contacts.
- Ask your Linkedin connections if they're connected with someone you're not. Sometimes people put their emails there.
- Github: A lot of software engineers and managers put their email there.

* Prepare a template -> unique to you with points you feel strongly about.
* Schedule an email for **8-9am in their timezone** (mon-fri), and wait for their reply. Make sure to follow up with a small ‘thank you’ message.
* This can be a **game changer** for 1st round interviews. Give it a try, see what all works for you, and experiment along the way! There’s no one who is fit for all - different things will work for different people.

University career fairs (Offline)

* All universities have career fairs once or twice a year which can be useful for making good connections with representatives from various companies. You’ll usually meet software engineers and recruiters from those companies, so it’s always a great way to put your best step forward.
* General tips: check out the companies coming, do your homework on them, prepare for simple coding questions, and be confident. Make genuine connections with people there, and have a fun time. Be **real**. Talk like a friend, ask smart questions, give them your resume, and sound interested even if you're not!
* Imagine yourself in their position - they're talking to hundreds of potential candidates, so how can you make yourself stand out? Prepare a background story, sound excited and interested (smiling helps), and have a normal conversation. Don't say things like "I'm a confident leader who would be great for the job".

- Eg: Career Fair Schedule | For Students | Career Development & Professional Connections

Online Career Fairs

- There are events happening across the globe where companies hire developers - another great networking opportunity.
- Some amazing examples:
    - GreyLock
    - Startup Fair
- There's an application process => apply, get in, and follow their process. It's a great way to get access to multiple companies and meet lots of amazing people.

LinkedIn job portal

- https://www.linkedin.com/jobs/
- It's important to have a good online presence. Linkedin is one of the more important online platforms, so make sure you have a nice, complete profile.
    - Add relevant experiences and projects, a small bio, and some relevant posts about what you're doing these days.
    - No one is staring at your profile for hours so don't spend too much time on it 😛
    - Add projects, get endorsed for skills, add strong headings and summaries. In general, most recruiters don't look at your summaries/bio/projects but it's helpful when you're applying through Linkedin.
- You can use the 'easy apply' feature on Linkedin and share your profile through that - make sure your profile is complete if you're looking to apply through this feature.

- We don't recommend it, because most people will apply through this - add some value by messaging a recruiter or an employee instead of applying randomly.

Referrals

- Referrals can get you a step ahead of others, leading you to a quicker first round interview (or even skipping the first round sometimes). Most big companies do this. This does not mean direct access to something, just that you'll get a response faster or lessen the amount of interviews you'll take.

- Developers working at companies often get incentives if someone joins the company from their referral, so there's nothing to feel shy about. Every engineer and manager likes a good candidate and it's a win-win situation for all.

- Flow: Send a simple 1-2 line message asking for the referral (with a selling point). Remember, be prepared before asking. The other person needs to actually like your profile because there's no obligation on their end to get involved. So make sure to add some skills, your online profiles, or a portfolio of projects.

Networking

- Connect with people from different websites, send them a message and try to see if they have anything to offer. This is a great way to meet founders/co-founders and ask if they're looking for new talent. They're always on a lookout for developers to join!

    - *"Hey person, hope you're doing well. It was great talking to you the other day about xyz. I would love to further connect with you on LinkedIn and discuss abc".*

    - *"Hey person, hope you're doing well. I was impressed by your website/company/project and would love to talk to you regarding some collaborations/career opportunities/inputs. Thank you"* - Works great, everyone likes to meet nice people.

- Events
    - Tons of events happen across the year where companies directly hire developers. Look out for those through your university portal, Linkedin, or other websites. Eg: Grace Hopper
    - There are career fairs as well, where company representatives gather and look for talents. Don't miss out on those if you're eligible to attend. It's always nice to meet new people and get informed about local opportunities.
    - Hackathons
        - I love hackathons - free food, awesome projects, amazing people, and wonderful ideas. Work on a project for 24-36 hrs straight and present your idea to win prizes. Meet like minded people and network with different company representatives who host booths there.
        - http://hackathonsnear.me/
        - https://mlh.io/

Discord, reddit

- Discord has amazing communities for skills, eg: You know Angular. Search for an Angular community on Google and join it. Help people, ask for projects, work for free, and make strong connections. Leverage these through your resume and mention it during interviews!
- There's also a job/contract/careers section where people are constantly looking for new contract developers. Message random people, get to know their projects/startups, and start working. Remote work has become popular and people are always looking for people to fill these roles.
- Eg: Angular Community on Discord

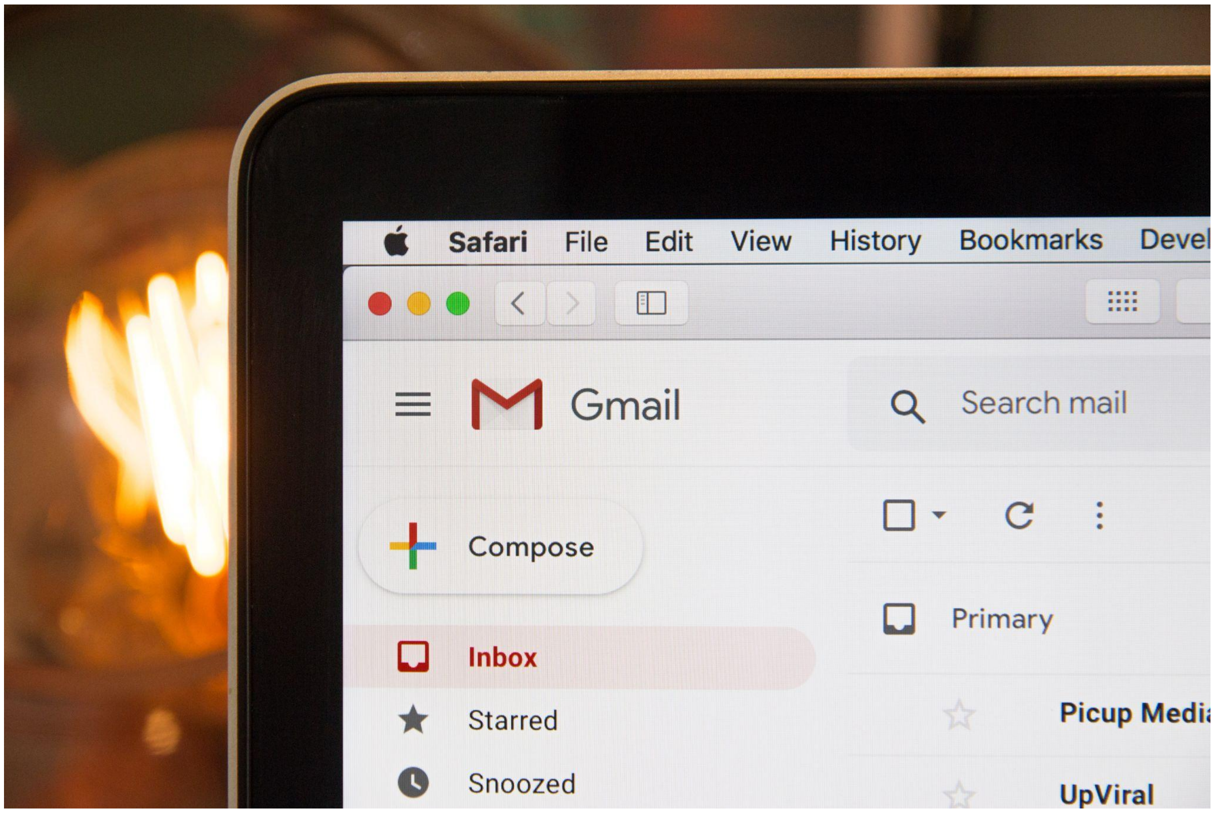
Online - careers website

- Companies have direct links to the job openings and it's usually under the 'careers' section on their site. So, just like Facebook (https://www.facebook.com/careers/) most companies have a designated page for their jobs.
- This is my least favorite way of applying and it should be the same for you. There are other better ways, mentioned below, which have a higher interview rate than this one.
- As an international student (in the USA), you should probably only do this as a last resort - most of the applications go through a screening process and it's harder to be noticed.

Remember

- You don't have to apply for 100's of positions to get an offer. Be PREPARED -> solve coding problems, make projects, and have fun. The ideal scenario should be to interview at 10-15 companies and, from this, get 2-3 offers. Work hard, but also enjoy the process. It is a long and tiring journey - make sure you take that in consideration. It's a marathon, not a sprint.
- If you're looking for a job/internship right now, you should understand one simple thing. It takes time. Be consistent and enjoy the process -> learn something new from this -> prepare for coding interviews -> and when you do get a job (which you will) -> give back to the community! You got this. We know **you're gonna kill it**, keep going! Good luck.

Cold Emailing, the right way



Cold emailing is one of the best skills to have in 2021. Here, we'll cover what you should do to get your next internship or job. This document will be targeted towards tech and computer science students, but it can also be useful for students in other fields.

Why try cold emailing? It works. We've used these simple tips and techniques to land interviews at Microsoft, Google, Dell, and many more big tech companies. By emailing a recruiter with the right content at the right time, you can land first round interviews at these amazing companies, and more.

Smart people always check their emails. So, if they find value in yours, they will reply and take things forward. However, it all depends on your email - which will be awesome after using the techniques discussed here.

Let's get started.

What is a cold email?

When you freeze your computer and send an email, it's called a cold email... Just kidding! A simple email to someone you don't know is a cold email. You can email any random person on this planet and, if they like your email, there's a high chance they will respond. This is how you build your network.

There are three parts of a cold email: the subject line, content, and value. The subject can be simple (nothing too fancy), but the content needs to be great. This will only work if you put in the work, you have the skills, and you are ACTUALLY very interested in what you're emailing about.

So, for getting a design internship, you have to both show *and* tell about the amazing work you've been doing. It's likely that if you tell someone you're just starting out and they should hire you as an intern, you will be ghosted. Don't do that. Do your work, make something awesome, and then add it there. This takes us to the next part, which is the selling point.

Selling point

Whenever you are sending an email, you have to keep your selling point in mind. Why should they talk to you? What value do you bring? How hard have you worked? WHY YOU?

It is important to think of different things you can show off about yourself, since you only have about 6-10 seconds to impress someone. Recruiters and managers are super busy, so it's crucial to consider the time frame and come up with something simple and catchy.

For developers, the selling point could be anything from a nice Github profile to a published project with actual users. It doesn't have to be something crazy and you don't have to be the best out there. It's important to show that you're working hard and actually building things.

Here are some examples of how you can use something your work to your advantage:

Project

I've been working on my app, which you can see here: Link. I've used the MERN (mongo, express, react, node) stack to develop this over the course of 3 months, and I can use these skills to my advantage if I get the internship opportunity.

GitHub

For the last 6 months, I've regularly updated my Github with multiple projects, coding questions, and class related stuff, which you can find here: Link. Most of these have been in react and node js, but I'm open to learning new things.

Hackathons

I've attended multiple hackathons in the last 6 months, creating over 4 mobile and web dev projects. You can find some of them on my Github here (Link), and I would be more than happy to share all more with you over a call or email.

Project

For the last 3 months, I've been working on a project called '30 days coding'. I'm helping students prepare for coding interviews by compiling the best free resources out there. We've helped thousands of students land internships and jobs in the last 6 months.

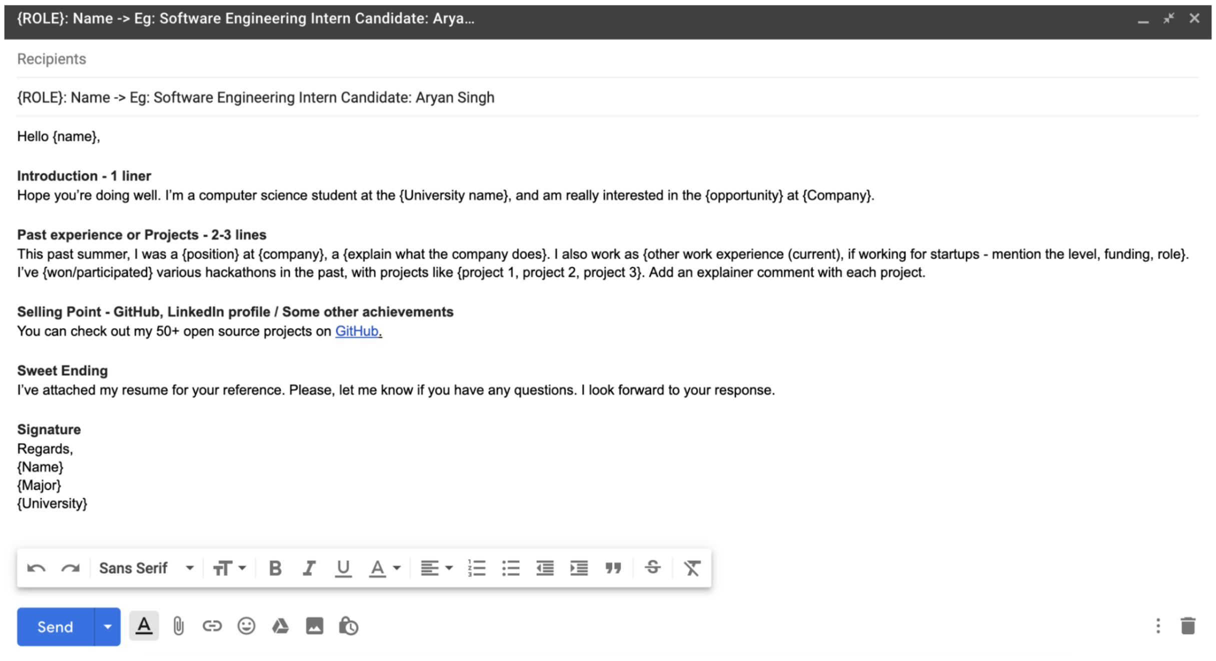
Classes

I'm a student, and, for the last 6 months, I've taken 3 more classes than the average sophomore student. Not only this, I've earned A's in all of them, and I'm continuing to work harder than before. I've also started gaining skills outside of school and would love to gain some more through the internship at your {company}.

Startup

I've been working with a startup called {name} where we've created a dashboard to find cat cafes near your location. We've had thousands of users since then and we aim to grow even more in the future. The mobile app uses react native, and I've learned about navigation, routing, architecture, and other amazing things that can be done with react native.

Templates


{ROLE}: Name -> Eg: Software Engineering Intern Candidate: Arya...
Recipients
Sans Serif
Send

{ROLE}: Name -> Eg: Software Engineering Intern Candidate: Aryan Singh

Hello {name},

Introduction - 1 liner
Hope you're doing well. I'm a computer science student at the {University name}, and am really interested in the {opportunity} at {Company}.

Past experience or Projects - 2-3 lines
This past summer, I was a {position} at {company}, a {explain what the company does}. I also work as {other work experience (current), if working for startups - mention the level, funding, role}. I've {won/participated} various hackathons in the past, with projects like {project 1, project 2, project 3}. Add an explainer comment with each project.

Selling Point - GitHub, LinkedIn profile / Some other achievements
You can check out my 50+ open source projects on GitHub.

Sweet Ending
I've attached my resume for your reference. Please, let me know if you have any questions. I look forward to your response.

Signature
Regards,
{Name}
{Major}
{University}

Here's a written sample:

Hey {**recruiter's name**},

{**Intro**}: Hope you're doing well. I'm a Computer Science student at {college} and am really interested in the **New-grad SDE position at Amazon.**

{Work experience}
This past summer, I was an intern at {} where I developed a web app using MERN tech stack. I've enjoyed developing mobile and web apps before, some of which you can find on my Github **here**.
My skills include: Frontend, Backend, Full-stack, Mobile, and Web development. I've had 4+ internships before with 10+ open source projects under my name. Here are some of the useful links for more information: **Github**, **Portfolio**, **LinkedIn**, **{skills and links}**

{**Ending message**}

I've also attached my resume for your reference. Looking forward to hearing from you! Thank you.

Regards,

Name

Linkedin, links...

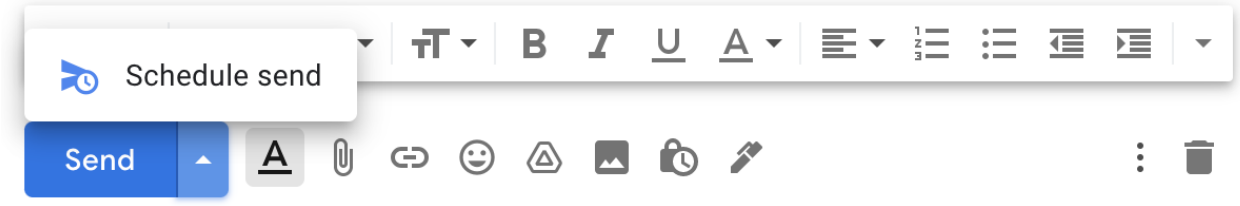
Some important takeaways:

- Add your links wherever possible.
- Add skills that the company uses. This could be something generic like mobile and web dev.

Timing

It's also important to time your emails. Here are some steps to follow

- Find the recruiter/manager's location.
- Schedule the email for the next working day's morning.



Subject

It's important to have a subject line and it's totally fine to keep it simple. Here are some examples:

- New grad 2021, Pinterest
- Software Engineer Intern 2021, Amazon

- College grad looking for New role

Including the recruiter's or manager's name would also make some difference. But, at the end of the day ,it's about what works for you. Try different things, see what works out, and then use that to succeed.

Sample Linkedin message

Hey! I'm a junior computer science student at UMass Amherst, and am looking for summer 2022 internships. I was an intern last summer at xyz and I've created abc or {something else here.}.
Please let me know if you can refer me for this position {link}. I've attached my resume for your reference.

Additional things you can add:

- Attach projects/links etc.
- Your leetcode stats

Resources

- 8 Cold Email Tips To Land Your Dream Job (With 3 Successful Examples)
- 8 Cold email templates
- How to write a cold email – Data Science Career Map
- How to write a cold email for job applications (with email templates) - 2021 update

---

How to take a technical phone interview

Read, research, and explore more to choose what's best for you! Let us know how we did @ 30dayscoding@gmail.com. Good luck, you'll do great!

Couple of days before your interview

- Go through a lot of easy questions to gain confidence on topics you're not-so confident about.
- Stop solving/preparing/studying for interviews, take a chill pill for 1-2 days. Understanding the concepts is fundamentally better than solving a couple of questions.
- Explore more about the company - teams, projects, languages, locations. You can do this on the company website, glassdoor, github, etc.
- Look at the **interviewer's profile** on linkedin. It can be useful to know where the person comes from and what their background is like. You can start by finding common ground, it's always nice to meet people with similar interests.
- Read solutions!
    - Reading questions or solutions doesn't mean just 'reading' them. It means understanding the core concepts used and considering how it is applicable to

other problems. So if you're reading a lot of DFS problems, then you'll start getting a sense of things whenever you see a new DFS problem. Also, you'll be confident + comfortable writing that!

Right before your interview

- Listen to music and pump yourself up! Relax, you got this. There are hundreds, if not thousands, of opportunities waiting for you if you don't get it. Don't stress yourself.
- Check your internet connection, keep your phone, copy, pen, pencil, and any other things you might need during the interview near you.
- Prepare a greeting + introduction message. This can be a compressed form of your past experience + projects + hobbies. It doesn't have to be long, but it should be something that will give the interviewer a sense of who you are and what you like.
- Prepare **questions to ask** at the end, in case the interviewer gives you time at the end. Examples:
    - What are the core company values?
    - How does a team function / how does your team work?
    - What kind of projects do you work on?
    - How has your time been there so far?
    - What would be some of my projects?

During your interview

Remember, you're going to talk to a human and be hired by a human, so it's better to have a conversation rather than robotic Q&A. Explaining the approach is much more important than writing the code. Writing code correctly is important, but its explanation must complement it. It's very very important to talk throughout the interview. There are multiple reasons for this, the most important being -> if you're stuck or going in the wrong direction -> the interviewer can help you! Here are some general tips that will help you more:

- **DON'T** code right away. Discuss the question with the interviewer. They are there to explain it to you, not fight or argue about something.

- Think, think, think, and **speak-out-loud** before starting to code. It's very important the interviewer knows what you're thinking so that they can help you! They can't help if they don't know.
- They're not looking at how fast you write code. Take it easy, type in slowly, speak while you're talking, and make sure they understand your overall solution.
- The interviewer will always try to help you with **hints**, lookout for those. Think about their reaction - positive or negative, and the move in that direction.
- There are 45-60 minutes. You **CAN** and **SHOULD** think for the first 5-10 minutes. Don't dive right into code and mess it up. Take your time and think out loud.
- Think of the time and space complexity when writing code, they'll always ask that after the question is done. If you don't know it or are confused, repeat the same drill - think out loud.
- Always come up with TEST cases when you finish writing code - "*let's test this with some edge cases*"

After your interview

- Follow up, even if the interview was bad. Write a simple email back saying how you enjoyed it and look forward to the next steps. It shows interest and everyone likes that. If the recruiter or manager replies, that's well and good. If they don't, you have nothing to lose.
- Make sure to check in after a week or two. A lot of times, recruiters forget to follow up so it's fine to send a friendly ping.

You're going to get what you're aiming for, very very soon! Keep working hard, you got this!

How to find coding projects + people

Read, research, and explore more to choose what's best for you! Let us know how we did @ 30dayscoding@gmail.com. Good luck, you'll do great!

Hackathons

- Hackathons are 2-3 day long events where you work in a team and make new projects. At the end, you present the project in front of judges and get feedback.
- It's a great way to find like-minded people, create something cool, and win awesome prizes! It's the best place to simultaneously find people and projects, create something new within 24-36 hours, and enjoy free food + swag from companies.
- The 8 Best Websites to Find a Hackathon — The HackLife Guide 💾 💻🤓

- How To Find Hackathons. One of the best experiences of my life... | by Felix Cabrera
- Tons of online and offline hackathons happening. Find, sign up, and attend: Major League Hacking (MLH).

GitHub + Open source

- Code Together · GitHub
- Clone and fork every repository you find useful. If you're interested in the project, track its progress or extend some features on it. For instance, there's a portfolio template -> extend it into something more centric to a topic. Try different things and keep exploring!
- Follow people who regularly add content. It's nice to see interesting tech on the Github feed. Once you follow people who post regularly, you'll see how so many amazing things exist in the tech world.
- **Be active overall**
    - Start with small projects.
    - Upload and maintain your projects on GitHub.
    - Add **documentation** wherever you can. Often, if someone is looking at your project, they'll only focus on the Readme. Not a lot of people will dig deep and find something in the code, but a lot of people are interested in what the project is about. So, it's important to add screenshots, write some about the project, and maybe explain the journey of how you made it.
    - Improve **Git** practices: Git is something that every company uses, and it's a great way to manage and collaborate on projects - definitely an essential part!
- The open source community is huge and it's great to follow these projects.
- A simple Google search of **"beginner project in react"** will land you at hundreds of repositories. Take advantage of this, clone the repos, and start learning from there.
- There's an *awesome* list for every framework out there - simply Google "*awesome list react*" and you'll see projects related to that. enaqx/awesome-react: A collection of awesome things regarding React ecosystem
- There are public repositories that bigger companies consistently manage - you can watch them and find an opportunity to contribute.

- Programs like **Gsoc** are absolutely amazing. Apply and try to get in - discuss with students who already had that experience.

Linkedin

- Tons of people post their projects on LinkedIn. Make sure to like and comment on their posts and message them about their journey.
- Post your project progress, ask for feedback, and connect with people who could work with you on it.
- Identify people who are generally active on Linkedin and reach out to them, learn more about what they're doing, and seek their advice. It's likely they'll help you or guide you to the right resources.
- LinkedIn Learning also offers some great project related learning: add a new skill set and take it from there.

Product Hunt

- Product Hunt – The best new products in tech.
- Product hunt has tons of professional products posted every day. Reach out to the founders and ask if they want to collaborate.
- You'll probably need skills relevant to that product, so make sure to do research and read about the product beforehand.

Simple projects

- Learn a framework, make simple projects, and work your way to something bigger. Start with popular frontend frameworks like React, Angular, and Django, then move to whatever you like more.
- There's tons of documentation out there. Google search or go on YouTube to learn the basics and create something small today. Add quick features, iterate it over time, and eventually make it big enough for others to join.
- Find problems around you and see if there is a simple solution. Top 10 Coding Projects for Beginners

- Find people once you have something mid-size. Ask for collaborations in a post on Linkedin, Reddit, Discord etc to find people.
- Look for other simple projects on GitHub, Reddit, Discord and ask if you can contribute. **Open source** is the way to go. We will make a separate guide on this.

Chrome extensions

- Create Chrome extensions to solve simple problems. I tried making a timer, note taker, etc for my own use, and learned a lot of new things.
- There are tons of new opportunities in this space. Start making small extensions and see how you can turn it into something big.
- Get started:
  - Build a Chrome Extension (Manifest V2) with React COMPLETE SOLUTION (2021 Web Development)
  - I Built My First Google Chrome Extension!

At your university

- Find other students who are in your classes. Tons of students are working on side projects. Just ask!
- Choose classes where you can make projects -> take the projects seriously and make them actually good. Continue to update them after the class is over.
- Clubs, events, and other small gatherings at your university may offer opportunities to make projects or work on ideas. Collaborate, network, and build something cool.

Upskill yourself

Upskill yourself -> At the end of the day, people love working with skillful developers. We're adding tons of resources on **30dayscoding.com**, feel free to look at those as well.

Here's the best full stack course if you're trying to learn web dev: Full stack open 2021. We are frequently adding these types of resources to our website. Keep learning!

How to learn a language/framework from scratch

Read, research, and explore more to choose what's best for you! Let us know how we did @ 30dayscoding@gmail.com. Good luck, you'll do great!

This document is mostly focused on frontend frameworks like React.

Introduction

- First few days of learning are an investment and will determine whether you stay with it or not. Most people leave right after the first few minutes or when they don't get their setup right!
- Read about what the framework is actually about - what problems it solves, how it is better than others, and how you can use it to the fullest. Read for a basic understanding.

Setup

- This is probably the most important step. It can sometimes be painful and annoying to set a path, install things, and finally get started with a basic app, but all of it is necessary.
- Things work out eventually. Watch a YouTube video if you're stuck, Google search the errors, and try to solve them. Don't stop before you get it right.
- You can also start with platforms like **Stackblitz** which we've discussed below.

Stackblitz

- DON'T set this up unless you're actually familiar with the framework. How can you become familiar with it? By simply going to StackBlitz: The online code editor for web apps. Powered by Visual Studio Code.
- Go to Stack Blitz, create a simple app, see how files are structured, and play around with it. Create some things and see the new changes. Add routing, make new pages, and create super small projectl.
- Google 'table example stack blitz' - you'll get hundreds of links of other people's Stack Blitz who have implemented those changes. This is super simple + effective. You can make even bigger things here. Although, creating something locally and then uploading to Github + Netlify has a better feel overall.

Learn the basics

- Start with the basics. Learn the building steps - how to add a button, how to change the text, etc. Try bigger things later.
- Read the **documentation**: there are multiple ways to implement a single thing and it's important to be aware of this. Reading the documentation is generally an important thing and will help you when building future projects.
- For example: Angular
  - HTML, CSS, Components
  - Input, output, different pages
  - Routing, transitions, passing data
  - Simple TODO app, couple pages

Tutorial hell

- Don't fall in the trap of just taking courses, watching tutorials, and copying code. Instead, be curious about problems and try to solve them yourself.
- Think about a **new feature**, Google how to do it, and then TRY it, instead of looking for courses/tutorials.
- Think of different **implementations** for the same thing. What can you do with firebase? What all can you do with a button and some URL redirection?
- One good approach is to find similar projects related to your problem. So, for instance, you want to make a page that has a card component and you want to replace it with something on a button click. Search how to make a quiz app. You'll find a similar approach that you can follow.
- Don't make a social media app. If you are following a tutorial which tells you to make one, close it, think of an idea similar to it, and try to make it by using what you learned in the course. Don't just blindly follow the same things that others are doing.
- https://www.youtube.com/watch?v=_3sPBX9TpyA&ab_channel=ChrisSean

Make a simple project

- Think of simple ideas, problems, and how you can solve them. Maybe you always wanted to make your portfolio website or something. Make it. Whatever you want to build: Make it. Start small and then go big.
- For example:
  - A website that allows you to find lost items
  - A landing page that displays products
  - Clicking inside a product to see more details
  - Extra: Add more features once you've completed the first ones
    - Add another page, smaller features
      - Add multiple users, authentication

Games

- Games are a great way to learn how complex things work. They're also a great way to understand OOPS. Most gaming classes will have gameOver(), start(), find(), and similar methods which help you write better code and understand how to avoid duplication.
- Most games also have a tricky gameOver() logic. For example Tic tac toe - you need 3 of the same things in the same line (in any direction). So, it's a good challenge to think about the logic before implementing the method.
- It's also exciting to see how frontend and logic work together when you're making games.

Bigger project

- Go to hackathons, find people, make connections, and work with someone. Working alone on a big project is the easiest path to not completing it. Therefore, you should find like-minded people, share your ideas with them, and finish a project together..
- Keep in mind your audience. It is good to make something that people might actually end up using! It could be anything from a lost-and-found website to something in education, travel, Chrome extensions, work life balance etc. There are tons of problems these days, so choose something and start building. It could even be as simple as https://30dayscoding.com if you like that stuff.

Don't make something for your resume - it won't last long.