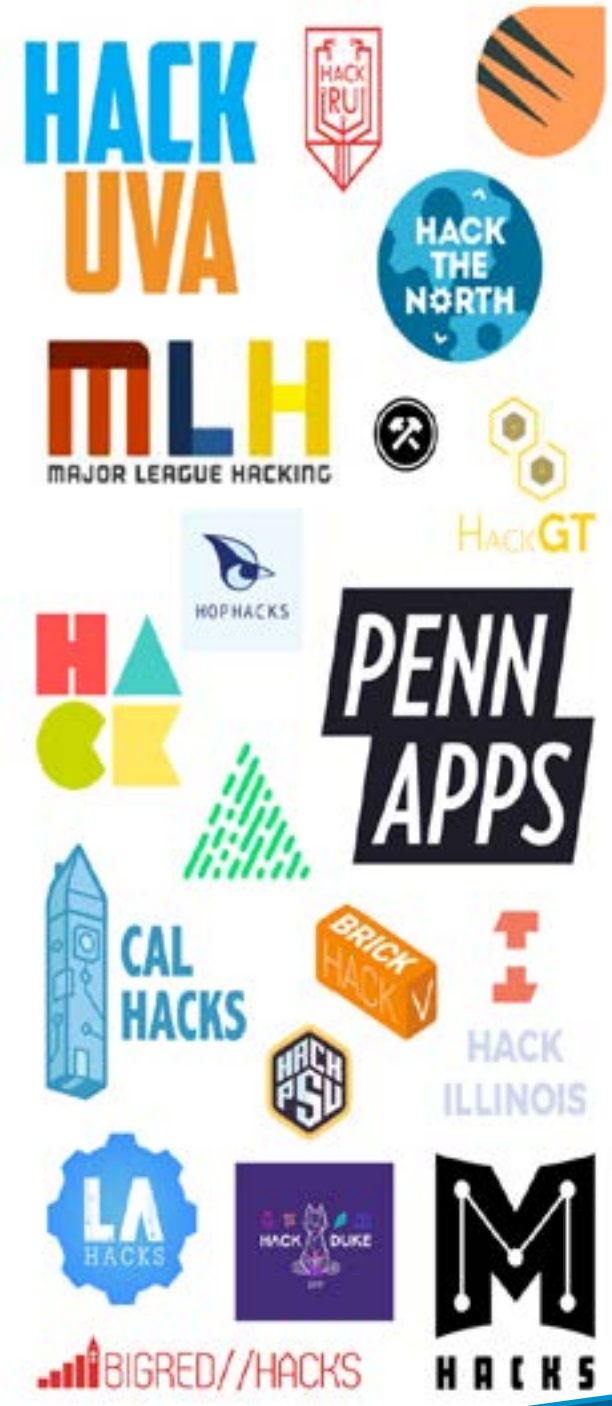
Nick Singh



Win Hackathons

A How-To Guide

Lessons Learned From Being On
Both Sides Of The Hackathon

Learn how to brainstorm the perfect hackathon project, execute on the idea well during the hackathon, and create a winning demo for the judges.

www.nipunsingh.com

What's inside this guide?

Chapter 1

Introduction

One of my favorite blogs is 'Both Sides of the Table' by serial entrepreneur and (now) venture capitalist, Mark Suster. As an operator-turned-investor, Mark has been on both sides of the negotiating table which has given him keen insight into the startup fundraising process and that's what makes his blog an entertaining, profitable read.

In March 2017, I found myself on "Both Sides of a Hackathon..'

Facebook, my employer at the time, sent me to Charlottesville to mentor and judge teams during the University of Virginia's 24-hour hackathon, HackUVA. It was an eye-opening experience being a judge. Mostly because the year before judging HackUVA, I had participated in the same event and won!

The year before judging HackUVA, I was a 4th year student at the University of Virginia and my younger brother was a senior in high school. At HackUVA, we teamed up to make DrawPlatformer, a game that bridges the gap between play in the physical and digital realms.

We were fortunate to win the grand prize (1st place) out of 35 submitted projects.

Finding myself a year later at the same hackathon, but as a judge now instead of a participant, gave me unique insights about what it takes to successfully win a hackathon. That's why I'm writing this guide today - to help you like how Mark Suster's blog has helped me.

After reflecting more deeply on the experience, I finally understood the rationale for why my team had also won 'Best Use of AWS' at PennApps in 2015, which (at the time) came as a complete surprise to our entire team.

Being on both sides of a hackathon also helped me pinpoint and articulate why my teams did **NOT** win Greylock Hackfest in 2016 or HackDuke in 2014. Getting to sit in the judge's room and deliberate over projects at HackUVA, I learned why some projects that seem good don't end up being winners.

a small caveat...

Of course, hackathons aren't all about winning. They can be a fun way to meet like-minded people and a place to learn and experiment with cool new technologies. If that's your main goal (which is 100% completely okay) this guide isn't for you.

But if you're a competitor, read on! Apply what you learn here, and you should be able to create impressive projects that are destined to **WIN HACKATHONS**.

Nick (Nipun) Singh

Chapter 2

Background

Before we start, here's a quick background on who this guide is meant for, and why winning a hackathon is worth all this effort.

Who This Guide Is For....

Hackathons are usually:

- 24-hour to 36-hour competitions
- competed in with teams of two to four people
- technology-oriented

This guide is primarily oriented at technical hackathons at the college level. Examples of these: MHacks, HackMIT, PennApps, HackInTheNorth, and HackUVA.

The tips in this guide can also be applied to science fairs, business plan competitions, pitch competitions, or case-study competitions, but this isn't my primary intention.

Why Even Desire To Win Hackathons?

If you don't win a hackathon, you leave with a cool project under your belt and a fun experience which is a perfectly okay outcome.

But if you do win, you can get:

- hired as a team to work at Nest Labs
- prizes like Nintendo Switches, Drones, and Raspberry Pi's
- cold hard cash
- an impressive story to tell during job interviews
- acquired by Apple
- [meta] the desire to write a guide on winning hackathons

Chapter 3

The Winning Idea

You can assemble a team of the best coders and still not win a hackathon if the idea you're working on is, as the kids call it, "lame AF."

So, before we talk about pitching your project, or how to execute effectively, let's address what makes some ideas more "win-worthy" than others. Learn why a good story is crucial, why demo-ability should be factored in when selecting ideas, and why NOT to chase API prizes.

The Idea Should Come with A Good Story

Your judges are human. Humans love to hear and embrace stories. Go read the Silicon Valley favorite 'Sapiens: A Brief History of Humankind' by Yuval Harari to understand how fundamental story-telling is to our species.

Pick ideas to work on for which you can tell a powerful story. A pain point you feel deeply is a great starting point because if you can keenly convey how excruciating and annoying the problem was to you and that you needed to solve it for yourself and other sufferers, that's one solid foundation! A good story creates buy in; the judges will empathize with your pain and will want to share your joy and relief.

Another source of ideas for a compelling story is sharing something unique about you or your team's personal background. If you're comfortable enough, maybe even open up about a hardship you faced which your technology might help solve. At PennApps, I saw a great project from a friend who suffers from depression. He built an app to better manage his mood and keep himself mentally healthy. His project was well-received by the judges and audience, and it all started with a good story.

How I Personally Applied the Above Tip to Win:

At HackUVA, I teamed up with my little brother who was a senior in high-school when I was a 4th year at UVA. I wanted to play to our uniqueness of being a sibling team. I wanted to play on the note that he was leaving soon for college and that I would soon be working a full-time job (#adulting). Regardless of what we made, I knew that just participating as a sibling team for my last college hackathon was a great story in itself. To emphasizes this, I knew we had to do something related to our shared childhood.

We brainstormed ideas involving Matchbox cars, Legos, Toy Soldiers, Play-Doh, & Pokémon Cards. These were all shared interests of ours from our childhoods. They symbolized having fun and playing together before video games, YouTube and other digital toys took over as our favorite forms of play.

That's why we settled on making DrawPlatformer, a game in which you draw a game on paper, and then our Computer Vision algorithm would turn it into a playable platformer-style video game.

We told the story of brothers who would play, draw and make art together. And we explained how our creative play in the physical world was essentially hijacked by the addictive digital world of Mario Brothers on our Gameboy. We told the story of wanting to bridge the physical and digital worlds, to have the fun of a digital video game but using the creativity we used to employ with physical toys. Our story resonated with many of our judges.

The Idea Should Demo Well

I've dedicated a whole section to creating and presenting the demo. But this is an important element to consider even during your idea-selection phase.

Go with an idea that makes the demo itself as captivating as possible.

- Can the demo be interactive?
- Is the demo visually interesting (think fireworks, flashing lights, flying drones, etc.)?
- Can the demo be funny?

An example of an idea that included a great demo was my friends' project at HackUVA 2018. They made FallSafe, an application that uses Computer Vision on live stream camera video to detect people who have fallen and calls the authorities. Their demo was hilarious!

They alternated between walking past the Nest cam normally and then hysterically falling on the ground and flailing in dramatic fashion. The judges did the same. It was quite the spectacle, and everyone was laughing. The team ended up winning 1st place in the 'Safety' track.

How I Personally Applied the Above Tip to Win:

For DrawPlatformer at HackUVA, we knew the idea would demo well because of the interactive nature of the product. We knew we could take a judge's hand-drawn creation and let them play it within 30 seconds. We knew it would feel and look like magic. Our demo amazed many people, so it was one of the deciding factors when it came time for the judges to pick the winners.

I'll expand in detail (and include photos of our setup) later in the 'Demo' section of this guide.

The Idea Should Be Technically Advanced

Some of you are probably thinking right now:
"NICK, WTF? THIS IS A HACKATHON GUIDE! WHY ARE WE TALKING ABOUT STORIES AND DEMOS? THIS IS A TECHNICAL COMPETITION—LETS TALK TECHNOLOGY!"

If you're among them, I encourage you to subscribe to my newsletter where I'll be explaining in greater detail about one of the biggest revelations I've had, not just about hackathons, but about startups and technology in general. It's about how large an influence story, marketing, and positioning play for 'tech startups' and technical hackathons and how the role of superior technology is smaller than we think.

But I digress... let's actually talk technology!

You want your projects to be technically-advanced. You know this already. One way of getting technical inspiration is by reading research papers and seeing if you can replicate them or do something similar. Another source is getting inspiration from advanced classes you're enrolled in. They often have a project component attached to it for master's students, and that's a great place to look for inspiration.

How I Personally Applied This Tip to Win:

I took Information Retrieval (the science behind search engines) and got the idea for making a humor search engine. That urge led to coming up with the idea for AutoCaption.co, an app that blazes through a corpus of quotes, jokes, and lyrics to deliver the most relevant caption for the user's photos.

DrawPlatformer was inspired by my Computer Vision class. I enjoyed playing around with the different OpenCV APIs and trying out different filtering techniques.

How I DID NOT Apply This Tip and, as a result, DID NOT Win:

At Greylock Hackfest, the elite hackathon sponsored by the top VC firm Greylock (early investors in Facebook, Dropbox, and LinkedIn), they brought in top-notch judges including hotshot Directors and VPs of Engineering at top SV startups from the Greylock portfolio.

We demoed a way to hail an Uber via text. Our rationale was that this would be useful in cases where you run out of mobile data but can still send SMS texts. The judges quickly dismissed our project as being too simple because we glued the Uber API to the Twillio API and that was about it.

There was no NLP, ML, AI, CV, IoT, or other over-hyped acronyms in this hack.

That year at Greylock Hackfest, 5 out of the top 10 hacks at were related to chat bots and NLP. A few of them weren't even that good, and could have just been 20 if-then statements masquerading as NLP. I couldn't believe that these exceptional judges would be swayed by the hype (it was 2016 and chat bots were all the rage). Lesson learned!

A Method for Evaluating Ideas

Now that we know some criteria for a good idea, it's time to evaluate all the ideas and select the best one. Make a spreadsheet and place each idea in its own row. Make each column the metrics we've discussed: How good is the story ? How well does it demo? How technical is it?

Finally, determine how feasible it is to successfully **make** by your team. Rate each project on the metrics and then go with the project with the highest average score.

One caveat: Don't go with the idea with the highest average if it is nobody's top choice. Design by committee leads to poor results. You want to tackle an idea when—and only when—at least one person is a true believer. Going with everybody's second favorite idea is disheartening and ultimately dangerous.

One solution, should you find yourself in the above mess of not agreeing on an idea, is this: follow the Bezos principle of disagree and commit. If one or two people are passionate about an idea, go ahead and agree to it. Enthusiasm is contagious. Chances are you'll learn to love the project as you work on it.

API Prizes: A Word of Caution

Hackathons are often sponsored by companies that are looking to promote their products to developers. To incentivize product usage and generate awareness, these companies offer 'API Prizes'—special awards deliverable only to teams that integrate their companies' specific API into their hackathon project.

If you're a beginner hackathon-er and don't think you can win the whole hackathon, it's okay to focus on winning an API prize.

But if you're trying to win the whole hackathon, be very careful about chasing API prizes.

Chasing API prizes can become a dangerous distraction. If you use an API which narrows down the type of ideas that can be built, it can be a lot harder telling a unique story with your hack.

Plus, when you go after an API prize, you might have more competition from other teams than anticipated. If you want to understand why competition is so deadly, go read Peter Thiel's Zero to One or Rene Girard's Theory of Mimetic Desire.

I'll talk more about the risks of API prizes in the Execution section of this guide.

If you still choose to go the API prize route, be sure to make a project that uses advanced features of the API or that uses the API in a unique way. Simply integrating it isn't enough.

How I Personally Applied This Tip to Win:

At PennApps in 2015, my team and I made AutoCaption.co, an app that barrels through a corpus of quotes, jokes, and lyrics to get you the most relevant caption for your photo. Amazon awarded us the prize for 'Best Use Of AWS', which surprised our whole team. We forgot we were even in the running; it was that much of an afterthought!

After judging HackUVA and talking with several API prize judges, I now realize why our team won several years prior at PennApps. Most of the people we were competing against were using simple AWS services like EC2 servers or S3 for cloud storage. A big part our pitch was tuning AWS-hosted Elasticsearch to retrieve information (search) for us through our content to make sure the content was both funny and relevant. The core usage of an obscure AWS service most likely caught a judge's eye and led to the win.

Chapter 4

The Winning Execution Plan

What good is an idea if you can't execute on it well? In this section, I share tips on how to make the most of your 24-hours so you can effectively turn your idea into a reality.

Assemble Your Team & Idea Together Beforehand

You know the rules: you can't write code for your project before the hackathon starts. That's not allowed and is unethical to boot. So, don't do it!

But do work beforehand to assemble a team and pick an idea. As you saw earlier in this guide, plenty of thought needs to go into picking a great idea. Start brainstorming ideas a few weeks in advance. Trying to come up with a good idea on your bus ride to a hackathon isn't a recipe for success.

Similarly, assemble your team beforehand. Make sure you have good chemistry with your teammates. Ideally, have a designer or talented front-end developer since a good UI goes a long way in hackathons. Make sure your team is in agreement about what to build before the hackathon starts.

Stick with Tried-And-Tested Tools

Don't experiment with new languages or web frameworks; doing so will only slow you down. This is another reason I mention ignoring API Prizes. If you've always used Heroku and aren't familiar with AWS, it's simply not worth your time to learn how to host with AWS to become eligible for the AWS prize. As mentioned before, you probably won't win the AWS prize for just hosting your website anyway, so don't waste time trying to learn new tools.

You only have 24-hours. This is not the time and place to try the Javascript-framework-of--the-month.

How I Personally Applied This Tip to Win:

I used Python & Django for every hackathon project, since these are the tools I know best. Most people can easily read and write Python, which makes forming a productive team easy. All my projects used plain-ol'-JavaScript with JQuery instead of React.js or Angular.js which also made creating a cohesive team easy.

Ship the Minimum Viable Product

MVP stands for Minimum Viable Product. You can learn more here if you aren't familiar with this term popularized by the book "The Lean Startup".

Prioritize what you'll work on so a MVP can be created in 24 hours. Don't get distracted by extra features or nice-to-haves. The best way to go about this is this: before starting the hackathon, make a list of all the features and rank them in order of importance and then make sure your team is aligned on the prioritization.

Ship your MVP faster by taking shortcuts. Examples:

- Don't deploy publicly if it's too much work; as long as it runs locally, that's usually good enough.
- If you're a beginner and haven't worked much with databases, it's okay to simply store data locally in a text file, or to not have data persist between sessions
- Don't bother having a login or account registration screen

Miscellaneous Tips

I'm far more efficient with two screens than I am with one. Bring an external monitor if that will help you be more productive. I'll discuss this more detail in the 'demo' section of this guide, but an external monitor can also help you better present your ideas to judges.

Skip the swag grabs, tech talks, and other fun events. If you're here to win, you can't afford to be distracted. Use the saved time to take a nap, especially for 36-hour hackathons. With some rest, you'll think better and be in a more pleasant mood during judging & demos (which matters and will be discussed later in this guide).

Chapter 5

The Winning Demo

You've picked a great idea and executed on it, it's time to learn how to pitch the project and create an effective demo.

Make It Interactive

Have a few test cases lined up that you know you can handle and that make your product look good. But also allow the option of making the input interactive. Let the judges use the app or product themselves. This helps engage them. Also, it shows that your product is real and not completely hard-coded to just a few test inputs.

How I Personally Applied This Tip to Win:

We had pre-made drawings & levels for DrawPlatformer to showcase the best examples of our tech in action. But we gave judges the option to draw their own game if they wanted right on the spot.

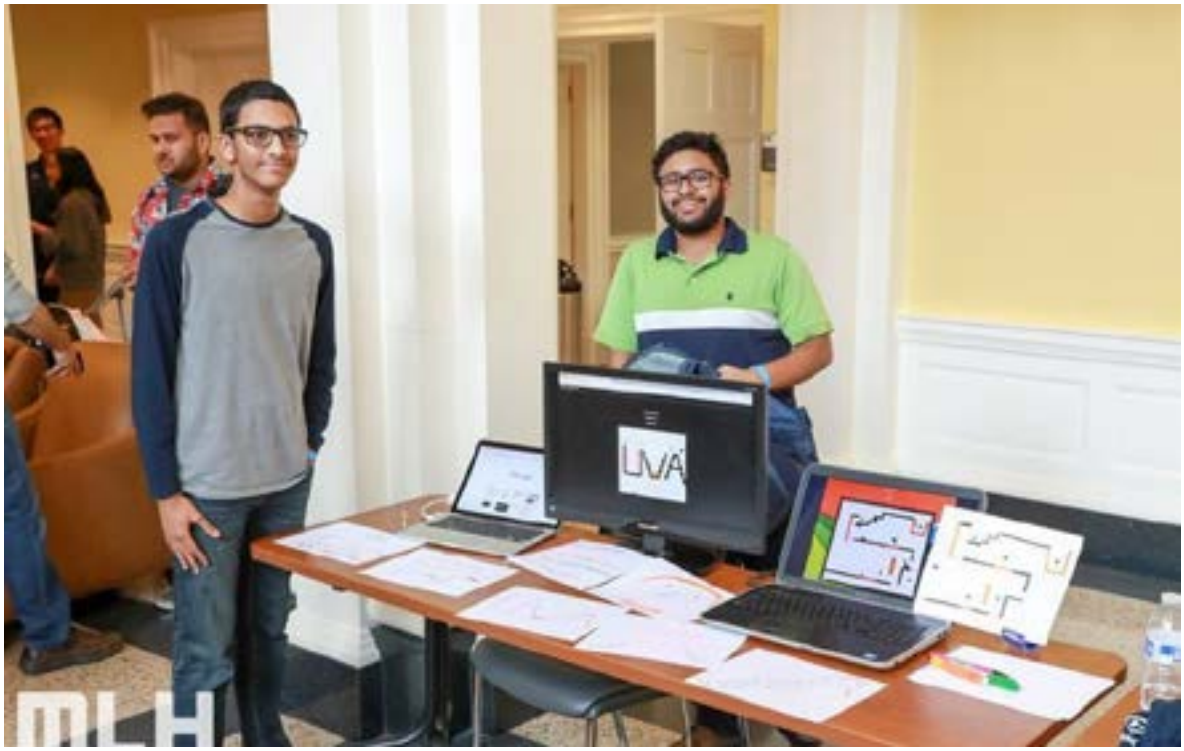
One judge was a big gamer. He was skeptical at whether what we promised worked, so he drew a game and it worked! He returned after the official judging to try and make a more complex game since he was so enthused by his first experience. I'm sure this went a long way toward helping us win.

Be Visually Creative

Figure out ways to make your project visually interesting (mentioned earlier in the ideas stage). This is why hardware hacks work wonders at hackathons—they're far more fun and visual to demo than is a backend project; you can't help but walk past some new contraption and stop to ask the team what it does. To make your software project captivating, get an external monitor to feature your product. Bring props if needed.

How I Personally Applied This Tip to Win:

Here's a photo of my brother and me presenting DrawPlatformer.



I showed the key steps we took in our Computer Vision algorithm in a video.

This was far more interesting than seeing some numbers or just abstracting away our CV algorithm and not showing off the technical work we did behind the scenes. This video was played on the left-most laptop.

On the external monitor, we had the game running for a 'UVA' level. On laptop #2, we had a more complicated level running with the actual hand drawing that created the game right by it to demo the power of our project.

We also had paper and markers so passersby and judges could make their own games. All this created a spectacle, attracting a lot attention and interest.

Plan for Murphy's Law

Murphy's Law is "Anything that can go wrong will go wrong". Plan for it. Have screenshots or video of your project in case it breaks ten minutes before the demo. Ideally, have the website or app running on multiple phones/laptops. Put your devices in **Do Not Disturb** mode so no embarrassing texts pop up while demoing.

Practice Your Varied-Length Verbal Pitches

Even though you have a few minutes to demo your project to judges, creating a more succinct version of the pitch is also useful. Condensing your pitch forces clarity. It makes you determine what's crucial and what's extraneous.

So, try to describe your project in a few words, in a tweet, and in a 2-minute pitch. This exercise helps you boil down your project to its essence, and then build back out the pitch.

Prepare Questions And Answers in Advance

Brainstorm the questions you think the judges will ask in advance. Then, answer them yourself and practice this Q&A. When judging happens, you'll feel more confident that your can address 90+% of the judges' questions.

This is also useful when you have a shy judge. You can feed them questions and control the narrative. Example: "So that's what my project does. Any questions? I'd be happy to talk more in detail about the computer vision behind the project or the motivation for why we even made DrawPlatformer". I give the judge a choice about what they want to hear, but it's not a real choice because I've prepared good answers to these two questions in advance and they're exactly what I want the judges to ask!

How I Applied This Tip to Win:

For DrawPlatformer, my brother and I quizzed each other as if we didn't know what the other person had worked on. We tried to poke holes. For questions where both of us could potentially answer, we pre-planned who would actually answer it.

This way, during judging time, we were completely in-sync. There were no awkward pauses to determine who would answer and no “Got’cha!” questions. For complicated questions—like “Can you explain the Computer Vision pipeline that takes in a picture and turns it into a game” —we had the video already pulled up and ready to go.

Be Maximally Approachable

I learned these tips during high school while competing in various science fair competitions at the state and national level. You want to be maximally approachable during the demo/judging time. Treat everyone as a judge and try to talk to as many people as you can. Keep smiling and looking approachable. Don’t fidget with your phone or exhibit negative body language.

When judges come by, be sure to introduce yourself, say your name clearly, and give a firm handshake. You’ll get back the energy and enthusiasm you put in. This works well, since the judges will advocate for your project with the same enthusiasm you pitched them.

Stay at your table for however long judging takes, even after your specified judges have finished judging you. API Prize judges might swing by. Other judges, who aren’t officially judging you, might swing by. These judges become important later on when determining an overall winner.

It’s also hard to know who’s a judge and who isn’t, given how many API Prizes and Partner judges there can be. So, just to be safe, treat everyone from start to finish like a judge.

I learned this while judging HackUVA. The initial two judges who were assigned to a project determined which were the few top projects they saw. Then, a larger group of judges got together to determine a winner for each track. The larger group of judges included people who hadn’t seen every project. That’s why, even if a judge isn’t officially judging you, you still need to impress them since they might become a decision-maker later in the process.

Chapter 6

Useful Reseources

This guide isn't exhaustive. Here's a shout out to a few other resources that can help you on your hacker journey.

The Major League Hacking calendar lists upcoming hackathons. They also have lists for Europe and Asia. Put these tips to good use and go compete!

Having trouble creating ideas? Use HackerEarth and DevPost to see projects that have been made at other hackathons. Go to the biggest hackathons and see which projects won. Study them to discover your own winning hackathon formula.

Product Hunt is another place to see cool projects, although many are full-fledged startups and not just weekend hacks.

Another place to get inspiration for ideas is YC's Request for Startups. It's tough to meaningfully solve any of the problems posed in the list over a weekend, but it does offer a place to start brainstorming.

Hackathon Hackers is a Facebook group that started out for hackathon-goers but is now generally a forum for anything tech-related. There still are sufficient posts about hackathons though.

Reddit can be another good place to find discussions on hackathons.

Chapter 6

Conclusion

This guide was long-winded enough - no need for a lengthy conclusion.
Just a heartfelt thanks for reading and I hope this genuinely helped.

Questions For You...

Did you successfully put these tips to use? Email me a link to your project—I sincerely want to see it!
Don't agree with these tips? Please email me with your feedback!

Join My Newsletter For More Guides

I send out a newsletter about once a month to 9,000 subscribers in 28 countries.
The newsletter covers topics such as Software Engineering Career Advice, Growth Engineering, Product Management, Strategy, and Startups.

Click here to join the newsletter!

About Me

I'm an engineer turned marketer, living in San Francisco. I'm currently running Growth at SafeGraph. Before that, I was a Software Engineer on Facebook's Growth team working on the Facebook Android app's user onboarding flows (New User Experience). I'm writing these guides as a way to share what I've learned with the community at large.

Let's Connect!

Connect with me on LinkedIn and on Twitter.