WIN HACKATHONS:
A HOW-TO GUIDE

Reflections From Being on Both Sides Of The Hackathon

Nick Singh
nipunsingh.com

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 sat on both sides of the negotiating table. This has given him keen insight into the startup fundraising process and is what makes his blog an entertaining and unique read.

In March of 2017, I found myself on *"Both Sides of the Hackathon"*. That's why I'm writing this guide. Facebook, my employer at the time, sent me to Charlottesville to help mentor and judge teams at the University of Virginia's 24-hour hackathon, HackUVA. It was an eye-opening experience - mostly because just the year before judging HackUVA, I had participated and won the same event!

In April of 2016, I was a 4th year 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 which bridged 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 in 2017 at the same exact hackathon, but as a judge now instead of a participant, gave me a unique perspective on what it takes to be successful at a hackathon. 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 had come as a complete surprise to the entire team.

Being on both sides of the 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 which seem good don't end up doing well.

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 (and that is 100% completely okay) **this guide is not for you**.

For those of you with a desire to compete - please read on. I hope you apply what you learn and go on to make impressive projects which will WIN HACKATHONS.

-Nick Singh

Nipun "Nick" Singh

TABLE OF CONTENTS

1. Background
    a. Who This Guide Is For
    b. Why You Want To Win Hackathons
2. The Idea
    a. The Hack Should Come With A Good Story
    b. The Hack Should Demo Well
    c. The Hack Should Be Technically Advanced
    d. A Method For Evaluating Ideas
    e. Note On API Prizes & Projects That Chase After Them
3. Execution
    a. Get Your Team & Idea Together Beforehand
    b. Stick With Tried-And-Tested Tools
    c. Ship The Minimum Viable Product
    d. Miscellaneous Tips
4. The Demo
    a. Make It Interactive
    b. Be Visually Creative
    c. Plan For Murphy's Law
    d. Practice Varied Lengths of The Verbal Pitch
    e. Prepare Questions In Advance
    f. Be Maximally Approachable
5. Additional Resources
6. About The Author

BACKGROUND

Before we start, some quick background on who this guide is aimed at, and why winning a hackathon is even worth all this effort.

Who This Guide Is For

Let's knock out the basics.

Hackathons are usually:

- 24-hour to 36-hour competitions
- competed in with teams of 2-4 people
- technology oriented

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

The tips covered in this guide could also be applied to science fairs, business plan competitions, pitch competitions, or case-study competitions but this was not the primary intention.

Why You Want To Win Hackathons

If you don't win a hackathon, you leave with a cool project under your belt and a fun experience. 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 money
- an impressive story to tell in job interviews
- acquired by Apple
- [meta] the desire to write a guide on winning hackathons

THE IDEA

You can assemble a team of the best coders and still not win a hackathon if the idea you are 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 winnable than others. Learn about why a good story is so important, why demo-ability should be factored in when selecting ideas, why not to chase after API prizes.

The Hack Should Come With A Good Story

Your judges are humans. And humans love to hear and believe in 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 craft and tell a powerful story. A pain point you deeply feel is a great thing to work on if you can strongly and emotionally convey how painful and annoying the problem was to you which you solved.

Another source of ideas for which stories are easy to tell is something unique to you or your team’s personal background. If comfortable, maybe even open up about a hardship you faced which your technology might help with. 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 and his project was received well by the judges and audience.

How I Personally Applied This Tip To Win:

At HackUVA, I teamed up with my little brother who was then a senior in high-school. I was a 4th year at UVA. I wanted to play to our uniqueness of being a sibling team. I wanted to play at the fact that he was leaving soon to college and that I was soon to start working a full-time job (#adulting). Regardless of what we made, I knew 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 childhood’s. Symbols of having fun and playing together, before video games and 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 on paper a game, and our Computer Vision algorithm would turn it into a playable platformer-style video game. We told the story of brothers who would play and draw and make art together. And how our

creative play in the physical world was upended by play in 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 with the creativity of how we used to play with physical toys. This story resonated with a lot of our judges.

The Hack Should Demo Well

I've dedicated a whole section to creating and presenting the demo. But it's an important aspect to consider even at the idea-selection phase.

Go with an idea for which the demo is interesting. Can the demo be interactive? Is the demo visually interesting (think fireworks, flashing lights, flying drones)? Can the demo be funny?

This is one reason why hardware hacks always do so well – there is just simply more to look at. A bunch of breadboards soldered together and connected to an Arduino visually looks more fancy and interesting than some Command-Line-Interface to a program you made.

An example for an idea which had a great demo was my friends' project at HackUVA 2018, who made FallSafe, an application which uses Computer Vision on live stream camera video to detect fallen people and then call the authorities. Their demo was hilarious – they literally just alternated between walking past the Nest cam normally and falling on the ground in dramatic fashion. The judges did the same. It worked and they won 1st place in the 'Safety' track.

How I Personally Applied This Tip To Win:

For DrawPlatformer, at HackUVA, we knew the idea would demo well due to 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, and was one of the big factors of why we won.

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

The Hack Should Be Technically Advanced

Some subset of readers 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”

To those readers, I highly encourage you to subscribe to my newsletter where in the future I’ll expand in more detail about one of the biggest revelations I’ve had – not just about hackathons - but startups and technology in general. It’s about how large of 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 talk technology!

You want your projects to be technically advanced. You knew this already. One way of getting technical inspiration is by reading research papers and seeing If you can replicate it or do something similar. Another source is getting inspiration from advanced classes you are enrolled in. They often have a project component attached to it for masters students, and that is 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 I got the idea for making a humor search engine. That urge led to coming up with the idea for AutoCaption.co – an app that searches through a corpus of quotes, jokes, and lyrics to get you the most relevant caption for your photo.

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 Personally DID NOT Apply This Tip So 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. These were hotshot Directors and VPs of Engineering at top SV startups from the Greylock portfolio.

We demoed a way to call an Uber via text. Our rationale was that this would be useful in cases where you run out of mobile data but could still send SMS texts. The judges quickly dismissed our project for 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.

A Method For Evaluating Ideas

Make a spreadsheet, and have each idea be its own row. Make each column the metrics we've discussed – how good of a story it is, how well it demos, how technical it is. Finally, also determine how feasible it is to actually make by your team. Rate each project on the metrics and go with the project with the highest average score.

One caveat: Don't go with the idea with the highest average but is nobody's top choice. Design by committee leads to poor results. You want to work on an idea for which at least one person is a true believer. Going with everybody's second favorite idea is dangerous.

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

Note On API Prizes & Projects That Chase After Them

Hackathons are often sponsored by companies who are looking to promote their products to developers. These companies, to incentive product usage and generate awareness, offer 'API Prizes' which are special awards only eligible to teams that integrate that companies' specific API into one's hackathon project

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

But if you are trying to win the whole hackathon, be very careful about chasing after API prizes. Chasing API prizes can be a distraction. It may be much harder telling a unique story with your project when you use an API which limits you to what kind of project that can be built. Plus, you'll have more competition as other teams who compete for that API inevitably will make similar projects to yours. That's why I recommend teams who want to win it all to focus on the bigger goal and completely ignore the smaller API prizes.

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

If you still choose to go down the API prize route, be sure to make a project that uses either advanced features of the API or 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 searches 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 realized why our team had won several years prior at PennApps. The bulk majority of 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 do information retrieval (search) for us through our corpus of 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.

EXECUTION

What good is an idea if you don'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.

Get 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 do. So don't do this.

But do work beforehand – on assembling a team and picking an idea. As you saw earlier in the guide, there needs to be plenty of thought that goes into picking a great idea. Start brainstorming ideas a few weeks in advance. Trying to come up with a good idea on the bus ride to the 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 aligned on what to build before the hackathon starts.

Stick With Tried-And-Tested Tools

Don't experiment with new languages or web frameworks as it will only slow you down. This is another reason I mention to ignore 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 anyways so don't waste time trying to learn new tools.

How I Personally Applied This Tip To Win:

I used Python & Django for every hackathon project since that's what I know best. Most people are able to easily read and write Python which makes forming a productive team easy. All my projects used plain-ol-JavaScript with JQuery rather than React.js or Angular.js.

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 work on so that an MVP is finished in 24 hours. Don't get distracted by extra features or nice-to-haves. The best way to go about this is before starting the hackathon, make a list of all the features and rank them in order of importance and make sure the team is aligned on this 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 are a beginner and haven't worked much with databases, it's okay to just store data locally in a text file or not have data persist between sessions
- Don't bother having a login or account registration screen

Miscellaneous Tips

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

Bring a power strip too – sometimes there might only be one outlet near your team.

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

THE DEMO

You've picked a great idea, and executed on it. Now 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 makes 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 to engage the judge.

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 on the spot.

One judge was a big gamer. It delighted him to no end to be able to draw his own game. He later came back, after 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 in helping us win.

Be Visually Creative

Figure out ways to make your project look visually interesting. mentioned in the ideas stage, this is why hardware hacks work so well. They are much more fun and visual to demo than a backend project. You can't help but walk past some new contraption and ask the team what it does. To make your software project more interesting, get an external monitor to better feature your product. Bring props if needed.

How I Personally Applied This Tip To Win:

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

I showed the key steps we took in our CV algorithm in a video of how we first fix the orientation, then we change the colors, then it gets mapped into pixels, then played as a game. This was much 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.

We also had paper and markers so passerby's and judges could make their own games. All this was a spectacle and attracted a lot attention.

Plan For Murphy's Law

Murphy's Law – "Anything that can go wrong will go wrong". Plan for it. Have screenshots or video of your project in case it breaks 10 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 awkward messages or notifications pop up.

Practice Varied Lengths of The Verbal Pitch

Even though you have a few minutes to demo your project to judges, creating a more succinct version of the pitch is also useful. Trying to condense your pitch forces clarity upon yourself. It makes you determine what is useful and what is not.

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 the project to its core essence.

Prepare Questions 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 confidently be able to address 90% of the judge's questions.

This is also useful in case 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 on what they want to hear, but it's not a real choice as I've prepped a good answer to both of these questions before-hand and it's exactly what I want the judges to ask about.

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 worked on. We tried to poke holes. We figured out, for questions where both of us could potentially answer, who would actually answer the question for the judges

This way, during judging time, we were completely in-sync. No awkward pauses to determine who would answer. No gotcha-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 when competing at various science fair competitions at the state and national level. You want to be maximally approachable. Treat everyone as a judge, and try to talk to as many people as you can. Stay smiling and look approachable. Don't fidget with your phone or have negative body language.

When judges come by, be sure to introduce yourself, say your name clearly, and give a firm handshake. You get back the energy and enthusiasm you put in.

Stay at your table for however long judging takes – even after your specified judges are done judging you. API Prize judges may swing by. Other judges, who aren't officially judging you, might swing by.

I learned this when judging HackUVA. The initial 2 judges that were assigned to a project determined which were the few top projects they saw. Then, a bigger group of judges got together to determine a winner for each track. The bigger group of judges included people who hadn't actually seen every project. That's why, even if a judge isn't officially judging you, you still need to impress them since they may become a decision maker later in the process.

It's also hard to know who is a judge and who is not a judge, given by how many API Prizes and Partner judges there can be. So to be safe, treat everyone from start to finish as a judge.

USEFUL RESOURCES

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. Go put these tips to use!

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 what won. Study them and 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 enough posts about hackathons though.

Reddit can also be a good place to see discussion on hackathons.

About The Author

I'm an engineer turned marketer, living in San Francisco. I'm currently running Growth at SafeGraph and before that was a Software Engineer on Facebook's Growth team working on the Facebook Android app's user onboarding flows (New User Experience).

When I was an undergrad at the University of Virginia, I interned as a Software Engineer at Microsoft and Google's Nest Labs.

Questions For You, The Esteemed Reader:

Did you successfully put these tips to use? Email me a link to your project – I want to see!
Don't agree with these tips? Please email me and let me know your feedback!
Want your money back? ~~Email me~~ This is a free resource!!

Join My Newsletter For More Guides & Tips

I send a newsletter about once a month to 8,000 subscribers across 23 countries on Software Engineering Career Advice, Growth Engineering, & Tech Strategy.
Click here to join the newsletter!

Connect With Me

Connect with me on LinkedIn and on Twitter.