Hackathon Tips

A compilation of tips to help you in hackathons.

Before the Hackathon

Attending your first hackathon can feel overwhelming, especially if you're new to building projects under time pressure. This guide outlines what to expect and how to prepare so you can focus on learning, building, and enjoying the experience.

What is a Hackathon?

A **hackathon** is a time-boxed event, typically 12–48 hours, where participants design and build a software or hardware project from scratch. Hackathons are commonly hosted by universities, companies, or tech communities.

- Participants work individually or in teams
- Projects are built during the event timeframe
- Events often include themed **tracks** (e.g., AI, sustainability, fintech)
- Teams present demos to judges at the end
- Awards are given for creativity, impact, or technical execution

Because of the limited time, judges prioritize:

- Original or compelling ideas
- Clear problem–solution alignment
- Functional demos with strong presentation

> A polished concept with a working demo usually scores higher than an ambitious but incomplete system.

How to Prepare

1. Define Your Goal

Before attending, decide what success looks like for you. Common goals include:

- Building a specific idea you're passionate about
- Learning a new technology or framework
- Networking with peers and sponsors

- Competing to win a prize

If you are joining as a team, align on a shared objective early. Misaligned expectations are a common source of hackathon friction.

2. Set Up Your Development Environment

Hackathon time is extremely limited. Environment setup during the event wastes valuable building time.

Before the hackathon:

- Install your IDE or editor
- Install required languages/runtimes (Node, Python, etc.)
- Install Git and setup your GitHub credentials
- Verify package managers work (npm, pip, etc.)
- Test a minimal project build/run locally

You may not write project code beforehand, but preparing your tooling is always allowed and strongly recommended.

3. Choose Your Tech Stack in Advance

Avoid debating languages or frameworks during the hackathon.

Select:

- Primary language
- Frontend framework
- Backend framework
- Database (if needed)
- Hosting/deployment option

The best stack is usually:

- Familiar to your team
- Fast to prototype with
- Easy to deploy

Hackathon Execution Strategy

Strive for Simplicity

Time constraints are the defining factor of hackathons.

Prioritize:

- A narrow problem scope
- One or two core features
- A working end-to-end flow

Avoid:

- Complex architectures
- Over-engineering
- Large feature sets

UI Matters More Than You Think

Judges typically see your project for only 1–2 minutes.

A strong UI:

- Makes the project immediately understandable
- Signals completeness and polish
- Improves perceived quality

Focus on:

- Clean layout
- Clear user flow
- Visual feedback or animations
- Demo-friendly screens

> A visually compelling interface significantly improves judging outcomes.

Use a Monorepo When Possible

For small teams under time pressure, a **monorepo** simplifies coordination.

Advantages:

- Single repository setup
- Shared types and models
- Easier deployment
- Faster onboarding for teammates

Frameworks like **Next.js**, **Supabase**, or full-stack templates allow frontend and backend development in one codebase, reducing integration overhead.

Collaborate with Branches and Frequent Merges

Parallel development is essential in short events.

Best practices:

- Each feature on its own branch
- Merge frequently into main
- Resolve conflicts early
- Keep main deployable

Avoid long-lived branches, integration issues compound quickly in hackathons.

Final Thoughts

Hackathons are intense by design. Expect fatigue, uncertainty, and time pressure, these are normal parts of the experience.

Focus on:

- Learning
- Building something functional
- Collaborating effectively
- Presenting clearly

Even if your project is incomplete, the skills and experience gained are the real outcome.

During the Hackathon

You're at a Hackathon. Now what?

1. Depth over Breadth

Hackathons are designed in a way where making a perfect product is impossible. As a result, getting stuck on small details or trying to perfect every functionality can cause more harm than good. Rather, try focusing on what *really* makes your idea stand out. What makes it different from what already exists? Why should the judges care about it? Prioritizing answering these questions and narrowing your scope can help polish what your focus and end result can be.

2. Learn While You Do

If it's your first hackathon, chances are your ideas may feel like they exceed your skill level. However, this is very normal, and that doesn't mean you should give up. Hackathons are also meant to be a very valuable learning experience where you can figure out what you already know and what you need to learn in order to accomplish your vision. Doing research such as watching videos, following tutorials, or reading documentation can help you take you through each step you get stuck on. Similarly, most hackathons may have mentors onsite specifically for support, so don't be afraid to ask for help.

3. Time, Time, Time

Hackathons are very short. They're usually over a single weekend, and most of the time you stay working overnight. By the end of the hackathon, you may feel drained or burnt out. To prevent this, it's important to pace yourself on what you want accomplished and plan ahead. If you're working in a team, it's very important that everybody knows who's doing what. As time passes, make sure that you have some sort of plan in mind that makes sure you stay on track. If you find yourself running out of time near the end, focus on polishing on a few key features that truly matter. And don't forget to take care of yourself-- taking breaks, eating or drinking water, or even taking a quick nap can help you function better when you truly need it.

4. Presentation Matters

Most hackathon judges won't look at your code. As a result, make sure that you can communicate what your project is trying to do and why it's important. Even if your project doesn't completely work as you wished, showing what the project is intended for and how it could work in theory can be just as important. As hackathons are so short, judges don't typically expect to see a complete working project. So, be sure to prepare a polished presentation, and leave a good amount of time near the end dedicated to making sure your project looks the best it can be.