

// video overview and full blueprint

How to build a Signal Project that gets you hired in 2026

The complete system for building a project that proves you solve
real problems, not just write code.

Watch the video first

This document is the overview for the "how to build a coding project that gets you hired in 2026" video. Watch the video, then use this as your map.

The steps below are the same ones covered in the video, condensed into a high-level walkthrough of the full system. This is the 30,000-foot view. As you move forward in the course, each phase gets its own deep-dive document with the exact, step-by-step process, templates, and examples for doing that part well. So treat this doc as your reference for the big picture. The deep-dive docs are where you actually execute.

Why this matters

Knowing how to code is no longer enough to get hired. When everyone can claim a skill, the skill stops being a differentiator. Companies are not looking for coders. They are looking for engineers who can solve real problems, understand users, build products that make the company money, integrate AI so the company does not get left behind, and see where things are heading in the next 5 to 10 years.

The only way to prove you are that kind of engineer, before a company ever hires you, is to show it. That is what a Signal Project does. It sends a signal that you think like a problem-solver, not just someone who can write code.

Here is the hard truth: most projects on most resumes prove the opposite. Tutorial clones, habit trackers, weather apps, hobby apps you built for yourself, and vague buzzword projects with no clear purpose all tell a company you are not that engineer. A Signal Project is the fix.

What a Signal Project actually proves

A Signal Project is a small, real product that solves a real problem for a specific type of company in a specific industry. It is not a full startup. It is a focused MVP you can build in one to two months. Done right, it proves the one thing companies actually care about: **proof of problem-solving, not proof of coding.**

Here is the full blueprint at a glance. The rest of this doc walks through each phase.

- 1 Target.** Choose the companies and industry you are building for.
- 2 Market research.** Build a company intelligence sheet and quietly start your network.
- 3 Problem selection.** Find the real trends and problems, then pick the right one.
- 4 Research camp.** Become a mini expert on that one problem.
- 5 Architecture.** Plan your MVP.
- 6 Build.** Assemble a small team and ship in one to two months.
- 7 Launch.** Treat it like a mini startup and create buzz.
- 8 Convert.** Turn the project into interviews and offers.

Phase 1: Target

Before you build anything, decide who you are building for. Tech is enormous, with thousands of segments: big tech, B2C, B2B, logistics, finance, healthcare, entertainment, startups, mid-sized, enterprise. You cannot build the right project, or land the job, if you do not know which companies you are aiming at.

Get specific on two things. First, company type and culture. A big company gives you a name brand but less impact and slower shipping, while a startup gives you speed, ownership, and cutting-edge work but more chaos and more hats to wear. Second, the industry that actually interests you. Everything after this phase depends on this choice, so do not skip it. If you are stuck, the course provides an LLM prompt to help you narrow it down.

Phase 2: Market research

Once you know your target, compile a list of companies you would genuinely want to work at. **No fewer than 10, no more than 100.** Keep it to companies in your industry, in an area you live in or would move to.

Build this as a simple sheet with five columns:

- Company name.

- ▶ Company LinkedIn.
- ▶ Relevant job openings, with the links pasted in so you can return to them.
- ▶ News on the company: funding rounds, layoffs and their stated reasons, new features, CEO interviews about what they are shipping next.
- ▶ Open source repositories: their public GitHub or Bitbucket, and the issues they are working on.

This sheet is your intelligence file. The news and the open source repos are where you will later spot the problems worth solving.

While you are in LinkedIn, do one more thing. Search the department names of the roles that are hiring, then connect with the people in those departments. **Do not message them yet.** You are quietly building a network of loose connections now, so that reaching out later feels natural instead of cold.

Phase 3: Problem selection

Now use everything you gathered to identify three to five key goals, trends, or problems these companies share. This is the foundation of your project and the thing that actually catches a company's attention.

Then pick one you can realistically build in one to two months. Counter to what you would expect, do not automatically go for the flashiest, most popular trend. Often the second, third, or fourth trend is the better choice. It is usually more achievable, and more importantly, it proves you understand the industry at a level that only real research reveals.

A quick example from the video: a bill of lading MVP built for logistics. Almost no one outside logistics knows what a bill of lading is, or that it is still mostly paper. Building something around that obscure but real problem pulled in serious attention from logistics startups and large companies, got more recruiter eyes, and even passed more ATS scans, all because that specific term signaled real domain understanding. That is the power of picking the less obvious problem.

Phase 4: Research camp

Once you have your problem, spend one to two weeks becoming a mini expert on it. Run a focused research sprint and answer questions like: Why does this problem exist? Why has no one solved it well? Why does it matter to the company, now and in the future? What solutions already exist, and where do they fall short? Are current

solutions expensive or cheap? This is what turns a coding exercise into something that looks and feels like a real product.

This is also where those quiet LinkedIn connections pay off. Message the people you connected with, but frame it correctly. You are not asking for a job or a referral. You are building this product with or without them, and you would value their expertise. Reference something real, like a talk their CEO gave or work their team is doing, and ask for their insight. People are far more willing to help when the message is genuine and not a pitch. You get real advice from people inside the industry, and months later when you apply, they already remember you as the person who built the thing.

Phase 5: Architecture

Now plan the MVP. Decide exactly how your product solves the problem: what the software does, what the UI looks like, whether it is an API or an app, and what the smallest valuable version looks like. The goal is to build something clearly valuable without disappearing for six months. YC companies routinely ship in a weekend, and you are building a focused MVP, so aim to ship in under a month.

Phase 6: Build

Ideally, do not build alone. Bring in friends, classmates, or people you meet online. A small, efficient team lets you build a bigger, higher-impact project than you could solo. It also means your teammates can land jobs off the same project, which grows everyone's network. Then build it, in one to two months, no longer.

Phase 7: Launch

This is where most people quit early, and it is where the real leverage is. Do not just build it and start applying. Launch it like a mini startup and put real effort in.

At minimum:

- ▶ A website that clearly shows what the product does.
- ▶ An open source GitHub repo so companies can see your code.
- ▶ A Medium article that walks through the problem, your research, why it intrigued you, and how you built the solution.

- ▶ Posts on Twitter and LinkedIn sharing the article and the story behind the problem. Post continuously, from build to finish, not just once.
- ▶ Relevant industry communities, like Facebook groups or school communities, where you can share the problem, offer the product for free, and get feedback.

You are not trying to go viral. You are trying to create just enough buzz, a few hundred GitHub stars or a few hundred Medium claps, to prove the product matters. That proof is what makes a company think, this person builds things that matter.

Phase 8: Convert

Only after you have launched do you start applying, and you do not just apply and hope. This gets its own in-depth document later in the course, but here is the short version.

First, go back to the people who responded during your research. Tell them you finished the product and are thinking about applying, and ask if their team or company is hiring. Some will hand you a referral or point you to the right person, which usually turns into an interview.

If few people replied, or no one offers a referral, apply to the company and then reach out directly. Find the right person on LinkedIn, use a tool like hunter.io to find their email, and send a genuine message referencing your LinkedIn connection and the product you built. Done well, this is how you turn a project into interviews.

What comes next

This overview is the map. From here, the course breaks each phase into its own deep-dive document with the exact process, templates, and examples for executing it well.

→ Move forward with the Signal Project steps in the next section.