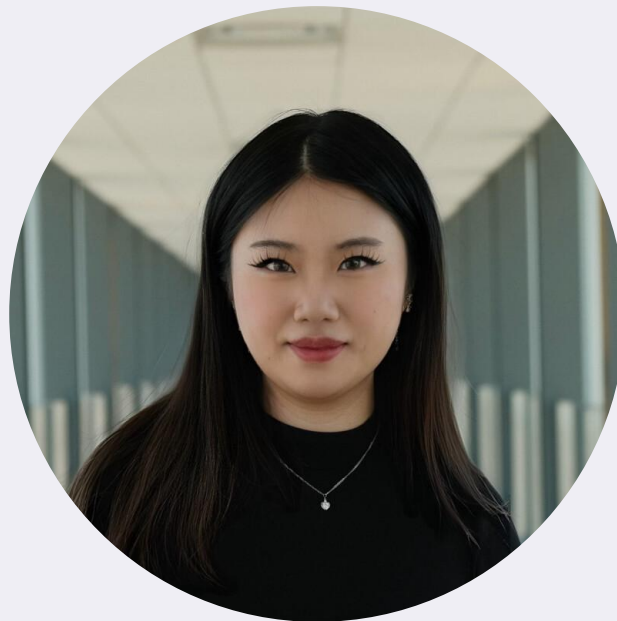



How to Start New Projects

Avery Lin - WiCS Orientation Fall 2025

A bit about me!

- Graduated in **June 2025** from CS
- Currently working as a **founding software engineer** at a SF-based startup
- Love going to the gym, playing video games, and matcha :)





✦

Table of contents

01 Why Projects

Why projects matter for learning and landing internships

02 Starting & Building

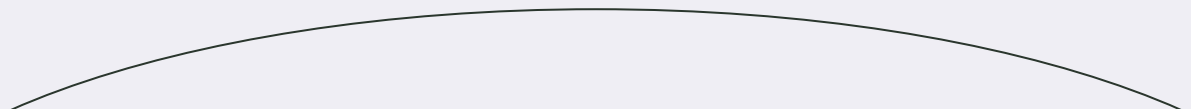
How to choose ideas that are simple, small, and personally meaningful

03 Sharing & Showcasing

A step-by-step approach to turning ideas into something that works

04 Final Thoughts

Summarizing what we've learned



✦



01

Why Projects Matter

What Is a Project?

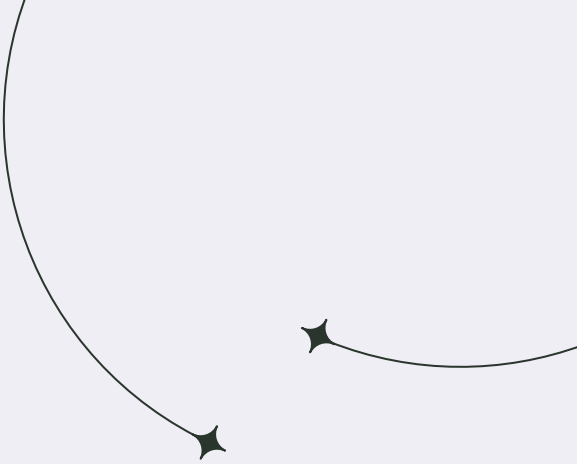
- A project is any program or system you create **outside of homework or internships**
- Even a small tool that solves one simple problem counts as a project
- It's your chance to apply class concepts in ways that feel practical and fun!



Why Build a Project?

- Projects give you **hands-on practice** with debugging, problem-solving, and finishing something real
- Each project adds to a **visible track record** you can actually show to others
- Recruiters and interviewers often use projects as **proof** of your engineering skills!





02

How to Pick & Start Building a Project

How to Pick a Project

- **Do projects you actually care about.** Think about everyday problems, hobbies, or classes that frustrate you – could you solve one with code?
- **Don't just build what looks cool to others.** A project you care about is far more likely to get finished (and teach you more) than one you picked just to impress
- **Start small!** Your first few projects should take no longer than a weekend to a week
- **Avoid giant projects** like “build the next Instagram” until you've built some smaller ones first



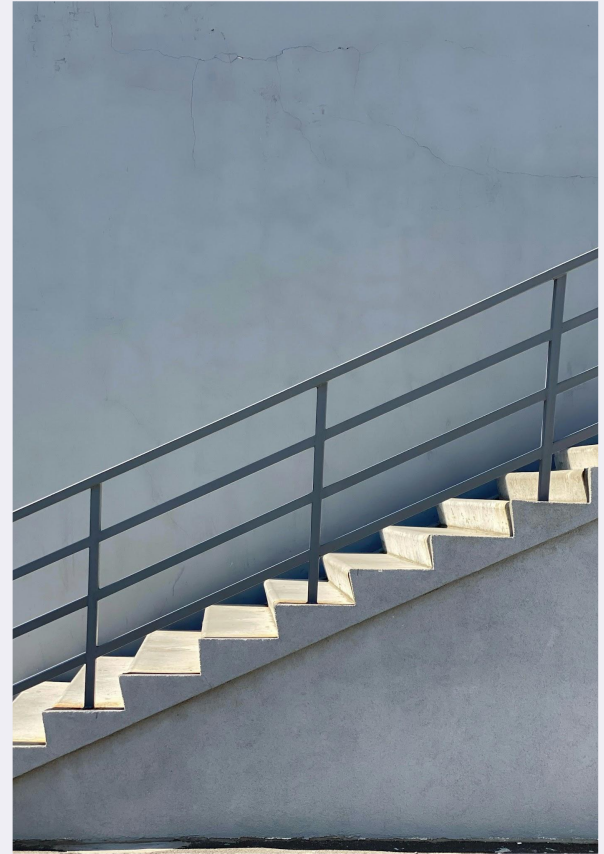
How to Pick a Project

- Use ChatGPT as a **brainstorming partner**
 - Try asking, *“What programming projects can I build to solve X problem in Y days as a first-year university student?”*
- Browse sites like [Product Hunt](#) for inspiration and see what small tools people are building
- Consider joining a **hackathon** – they provide prompts and constraints that make choosing a project much easier!



Starting Small

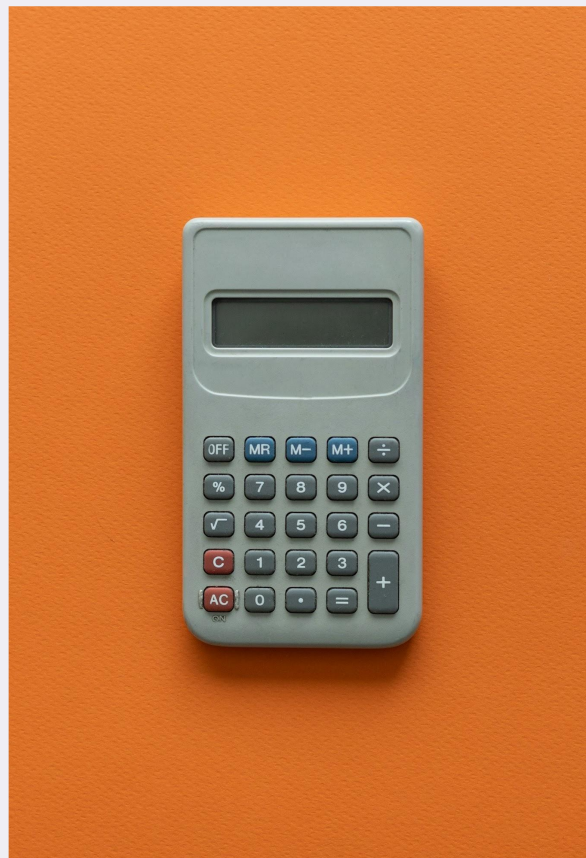
- Start with the **smallest version of your project** that actually works, and ignore everything else until that's done
- Write down the first tiny goal (for example, "make a button appear on the screen"), **finish it**, then move to the next
- Once something works, add the **next small improvement**, and repeat until the whole project is complete
- If you get stuck, shrink the goal into something even **simpler** – for example, just making the button show text before making it do anything on click



Building a Calculator

Step-by-Step

1. **Basic:** Make it handle two numbers with the four operations (add, subtract, multiply, divide), create a simple interface so you can type numbers and see the result on screen
2. **Improvement:** Allow decimals, add a clear/reset button, and show an error if someone tries dividing by zero
3. **Nice-to-have:** Add memory buttons (M+, M-, MR), calculation history, and eventually advanced features like parentheses or square roots



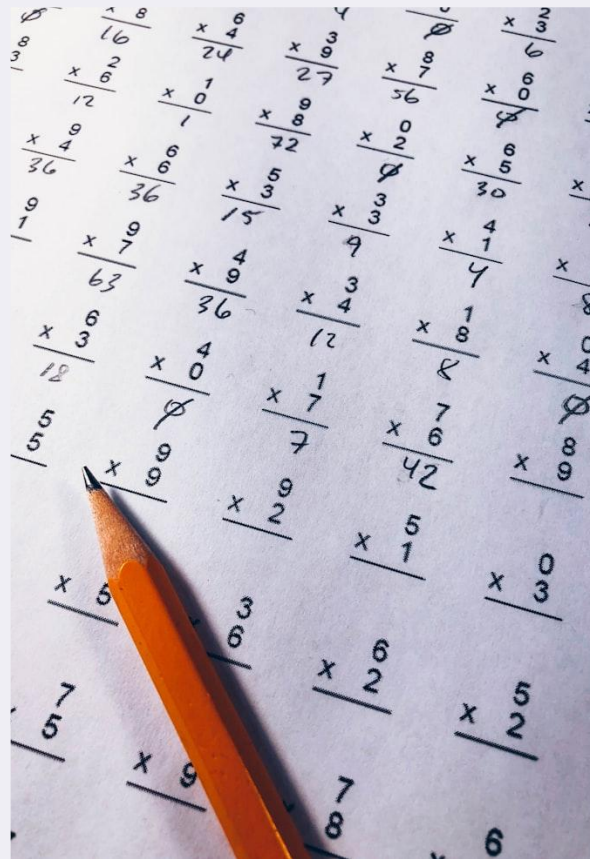
Getting Unstuck

- **Getting stuck is normal:** every programmer hits bugs and errors!
- **Use AI responsibly:** ask it to explain error messages or walk through your code step by step
- Build and test in **small steps** so you know which change caused the problem
- Search online for the exact **error message** – most issues you'll see have already been solved before



Step-by-Step Guide to Debugging

1. **Read the error:** Suppose you try $2 + "2"$ and see `"TypeError: unsupported operand type"`
2. **Find the problem:** One input was a number (2), the other was text ("2")
3. **Test smaller cases:** Try $2 + 2$ - now you see why the error happened
4. **Ask for help if needed:** Show the error, the code line, and what you tested - this makes it easier for others to guide you!



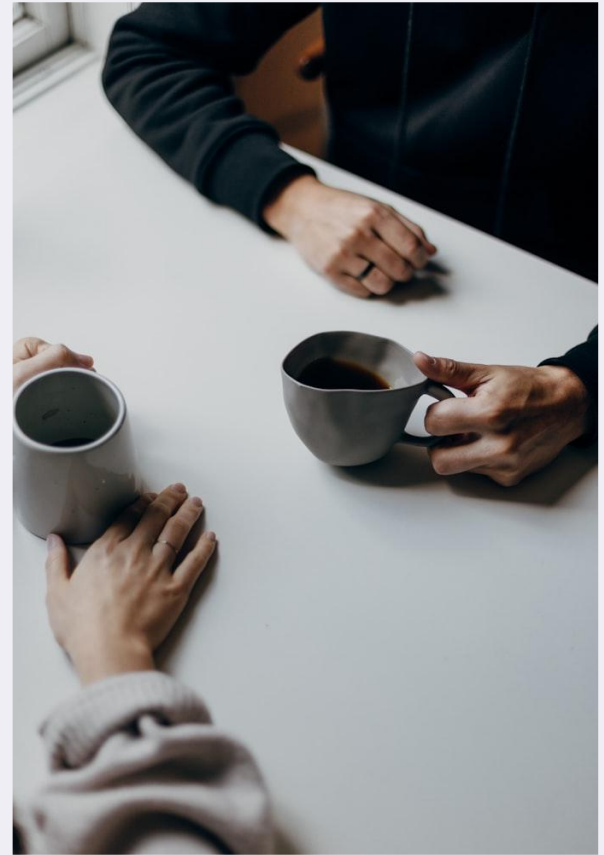


03

Sharing & Showcasing Your Project

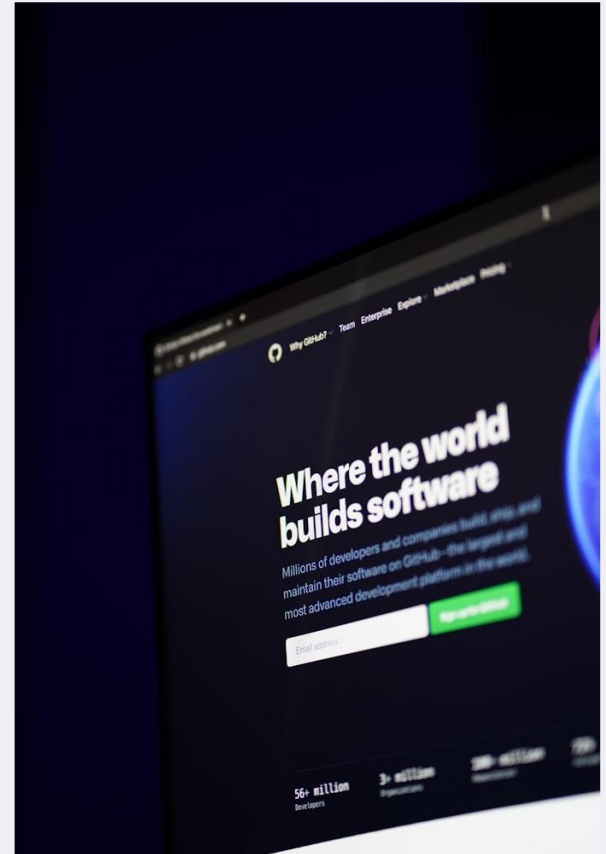
Why Sharing Matters

- Sharing your projects gives recruiters, interviewers, and teammates a clear way to **see your skills in action**
- It creates easy conversation starters – people can ask about what you built, how you built it, and what you learned.
- Even having small, finished projects online show steady **growth, initiative, and desire to learn** beyond classwork, which is a strong signal to future employers!



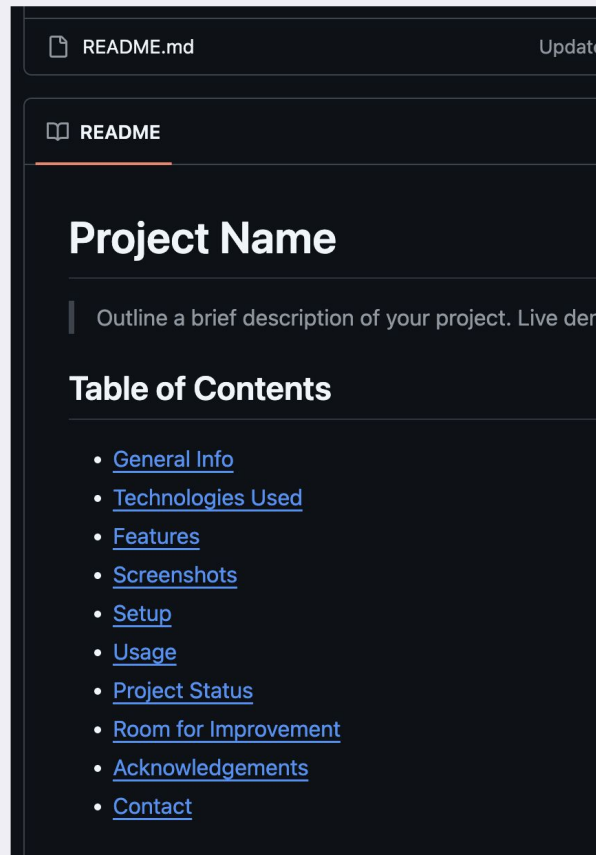
Using Git & GitHub

- Git is a tool that lets you **save checkpoints** of your code so you can go back if something breaks
- GitHub is a website where you can upload your projects – it's like a **portfolio** for programmers.
- Start with two simple commands:
 - `git add` → choose what code to save
 - `git commit` → write a note and save those changes on your computer
- Use `git push` to upload your changes to GitHub so your project is visible to everyone!



README

- A README is the **cover page** of a project on GitHub – it's what most people see first when checking out your project
- It should explain in **plain English**: what the project does, how to run it, why you built it, and what you learned
- Write it so that future employers, classmates, and coworkers can **quickly understand your work**, even if they don't code
- A clear README shows that you can **communicate**, reflect on what you learned, and share your growth, not just write code!
- Here is a good template to use!





04

Key Takeaways



Start Small, Finish Strong

A tiny completed project that you care a lot about is more powerful than a giant unfinished one



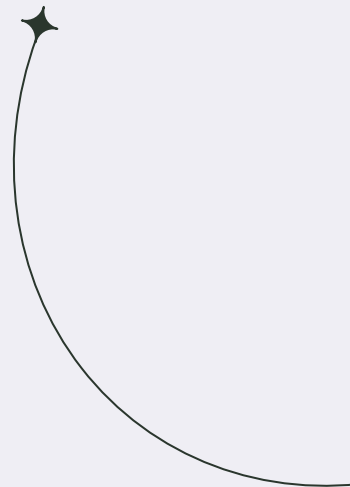
Build Step by Step

Add the necessary features before any nice-to-haves, and learn to test and fix along the way



Don't Chase Perfection

Every project will break – what matters is that you keep going, fix, and learn as you build





Thank you!

Any questions?